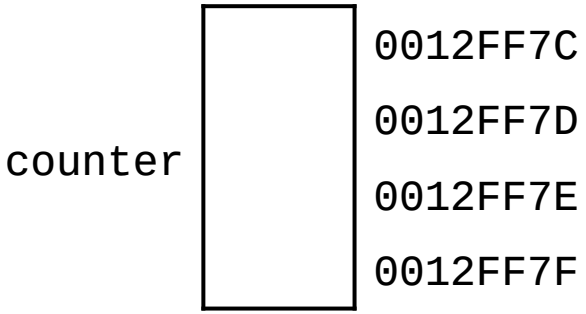
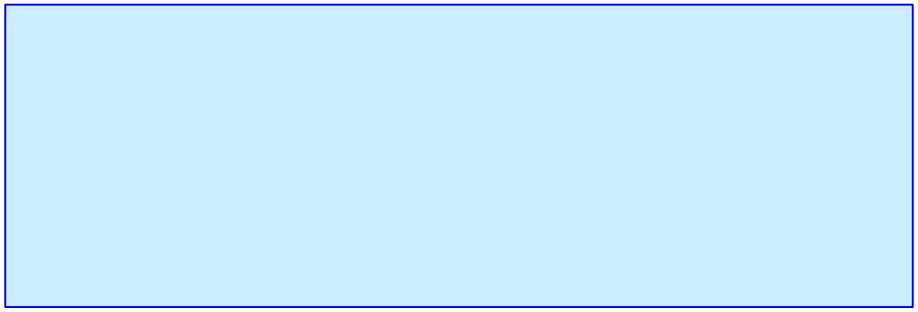
Ch. 2 Loops

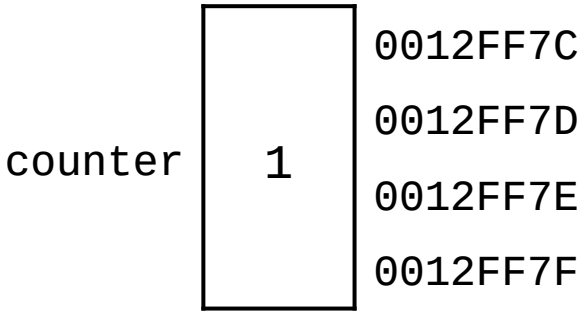
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



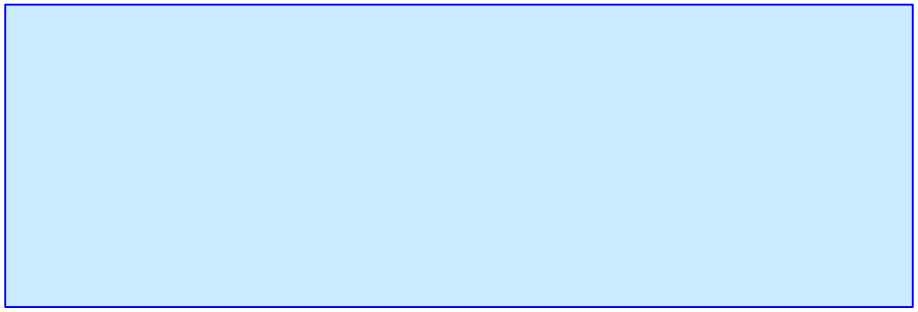
Output



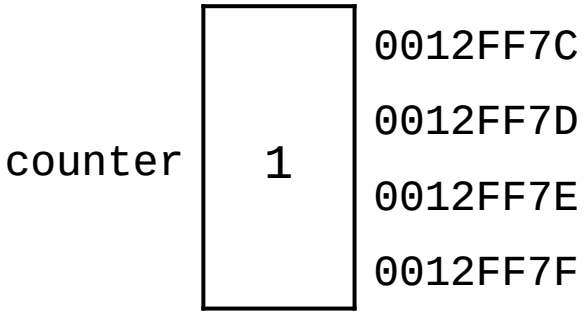
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output



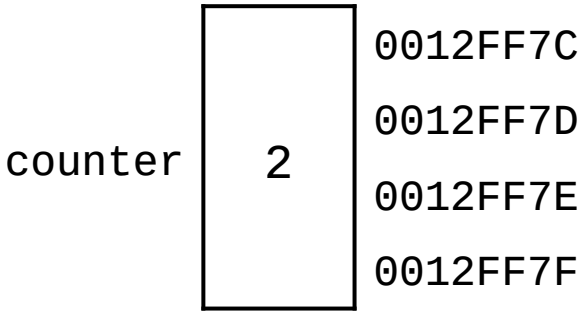
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1
```

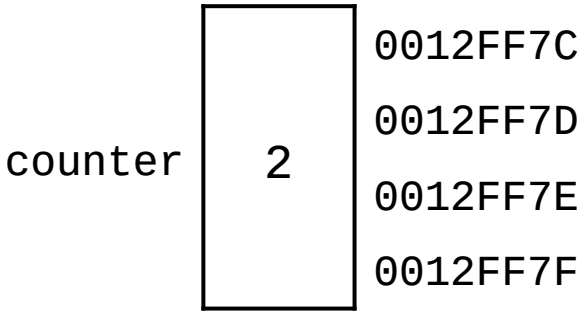
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1
```

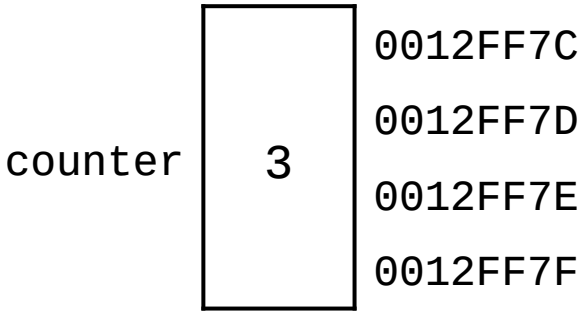
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1 2
```

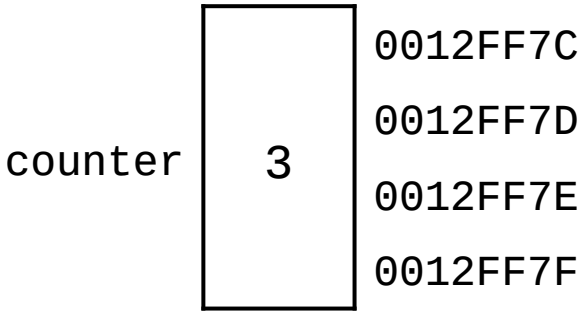
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1 2
```

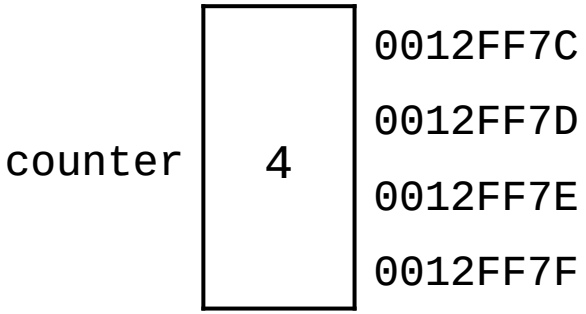
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1 2 3
```

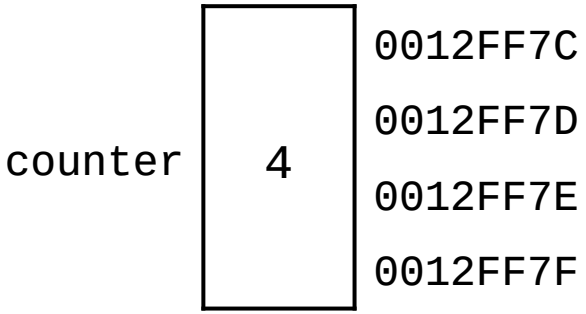
```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```



Output

```
1 2 3
```

```
int counter{ 1 };
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << counter << ' ';
++counter;
cout << endl;
```

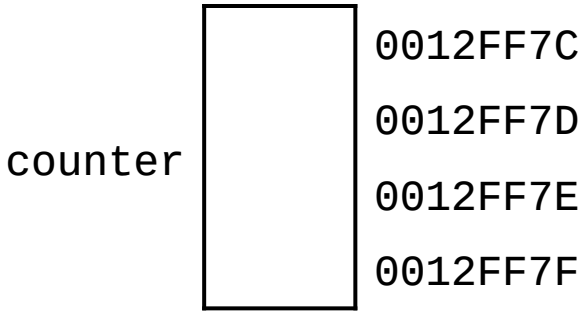


Output

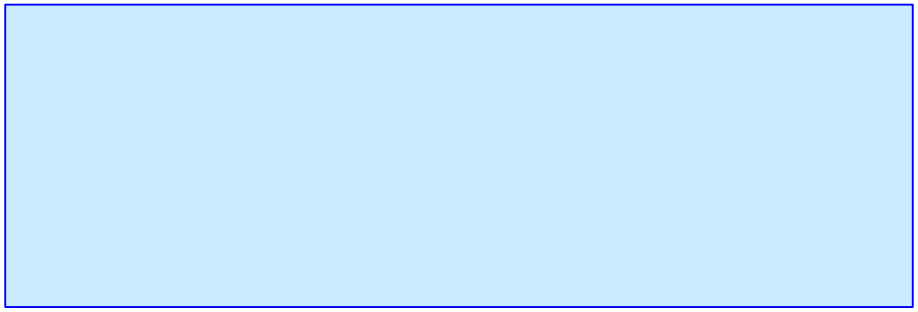
```
1 2 3
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

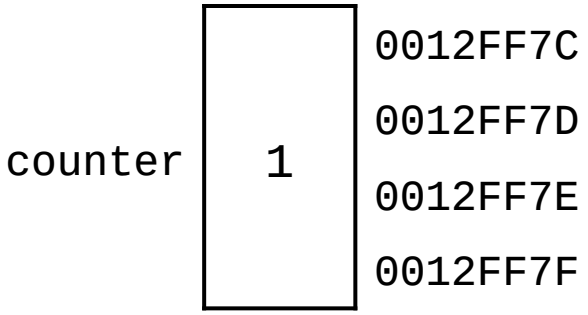


Output

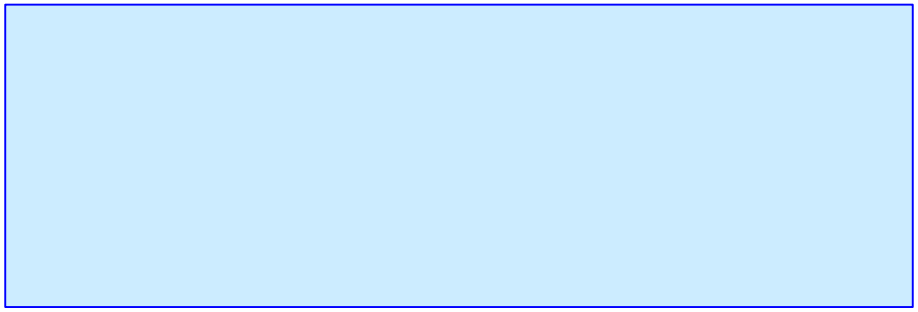


For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

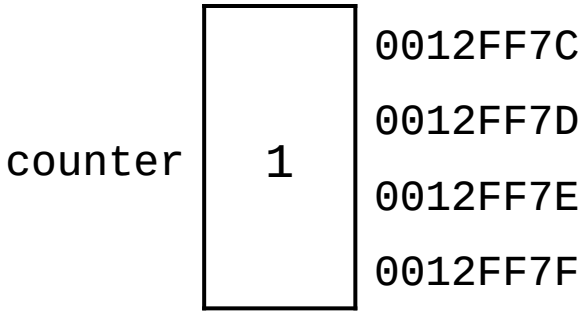


Output



For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

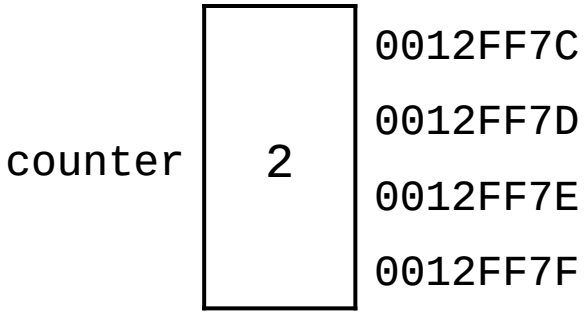


Output

```
1
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

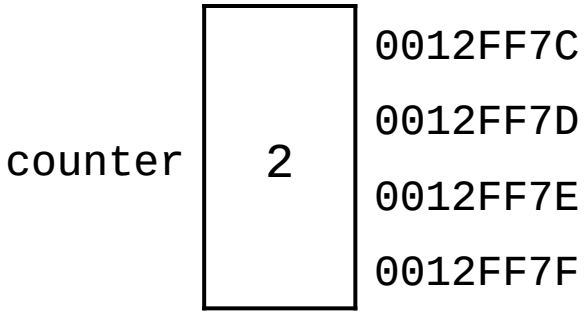


Output

```
1
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

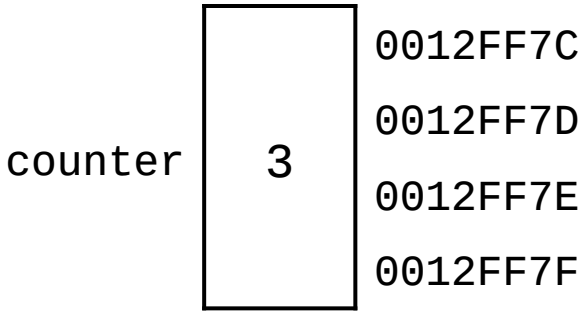


Output

```
1 2
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

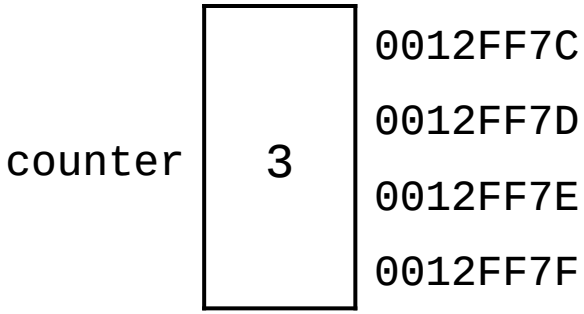


Output

```
1 2
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

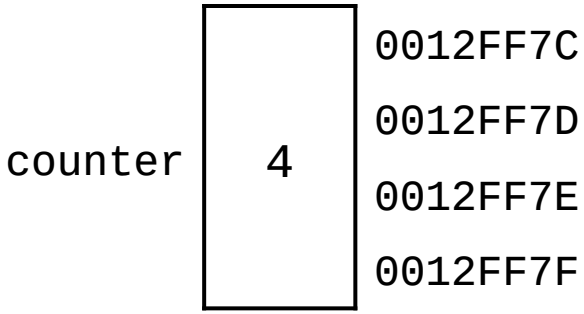


Output

```
1 2 3
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

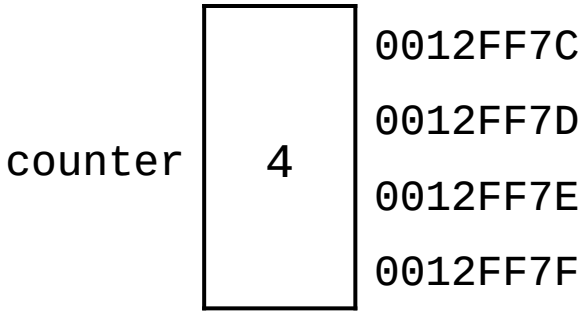


Output

```
1 2 3
```

For Statements

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```



Output

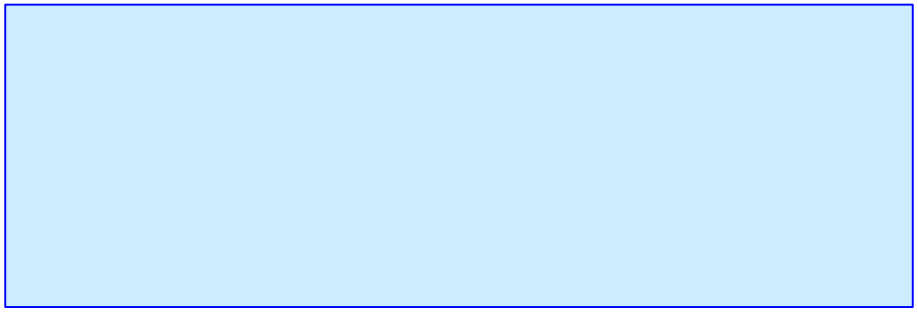
```
1 2 3
```

```
#include <iostream>
using namespace std;

int main()
{
   for( int counter{ 1 }; counter <= 10; ++counter )
      cout << counter << ' ';

   cout << endl;
}
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output



While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output

While Statements

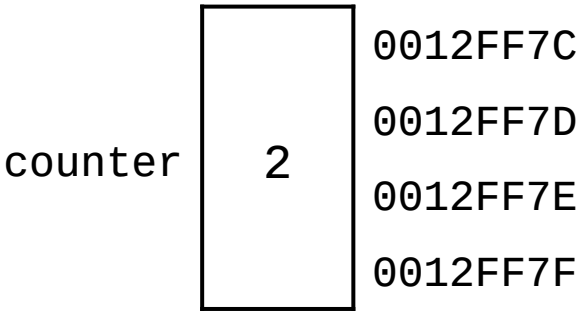
```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output

```
1
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

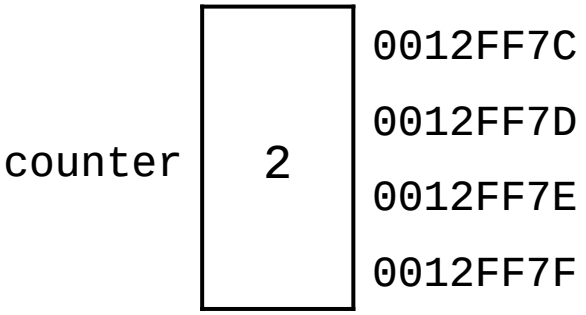


Output

```
1
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```



Output

```
1 2
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output

```
1 2
```

While Statements

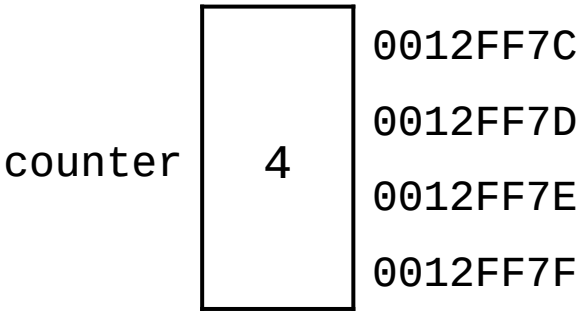
```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output

```
1 2 3
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

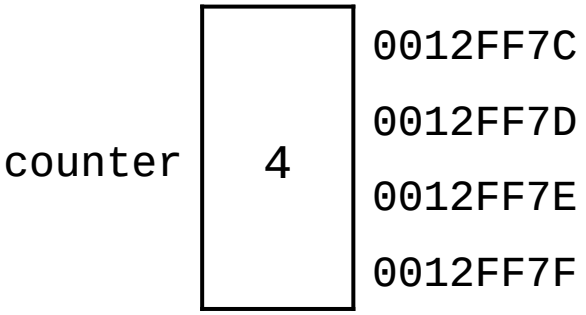


Output

```
1 2 3
```

While Statements

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```



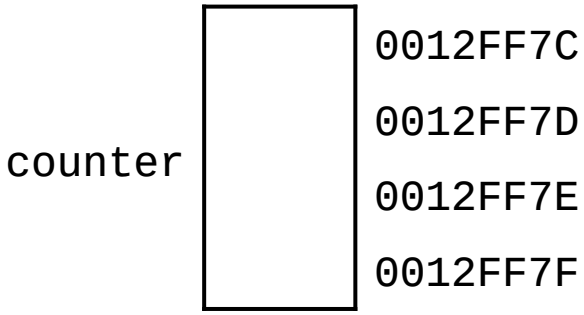
Output

```
1 2 3
```

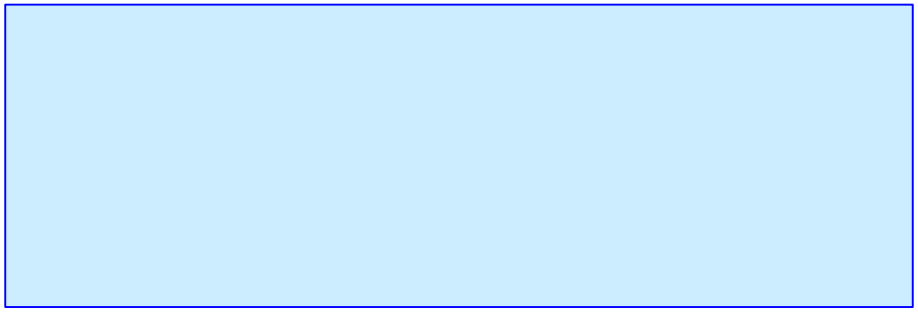
```
#include <iostream>
using namespace std;

int main()
{
   int counter{ 1 };

   while( counter <= 10 )
   {
      cout << counter << ' ';
      ++counter;
   }

   cout << endl;
}
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

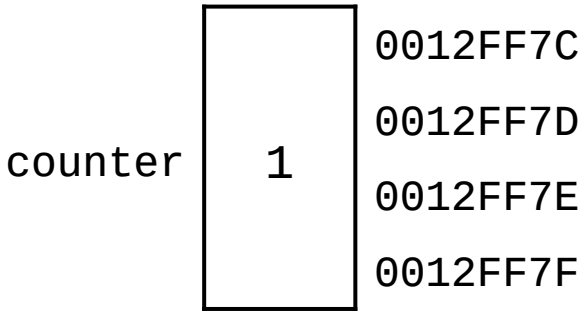


Output



Do While Statements

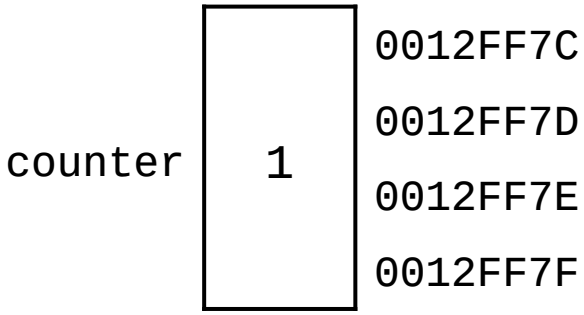
```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```



Output

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

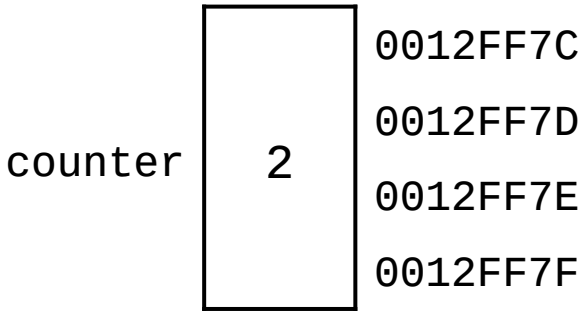


Output

```
1
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

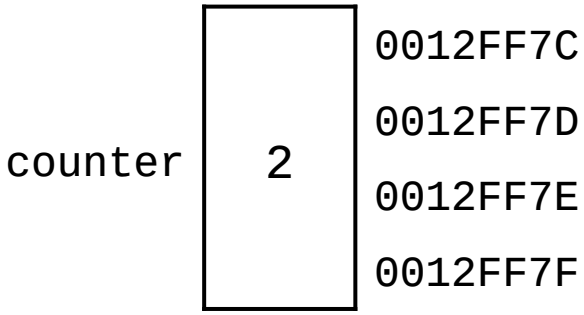


Output

```
1
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

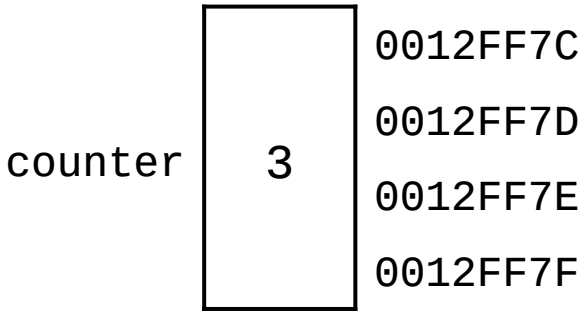


Output

```
1 2
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

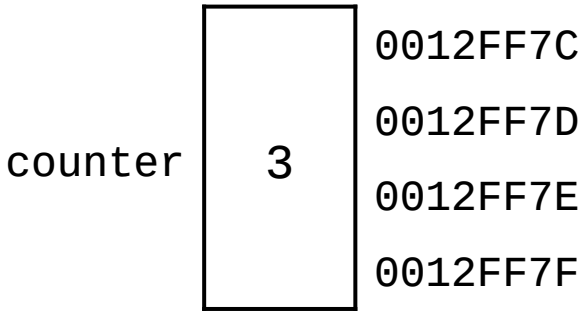


Output

```
1 2
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

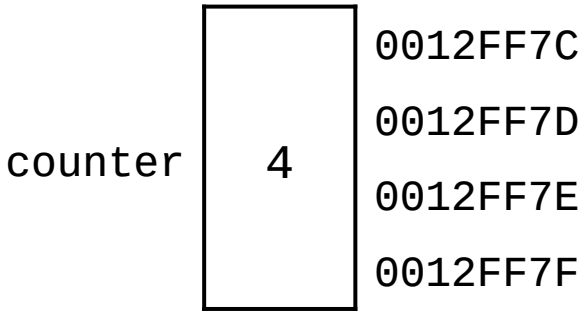


Output

```
1 2 3
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

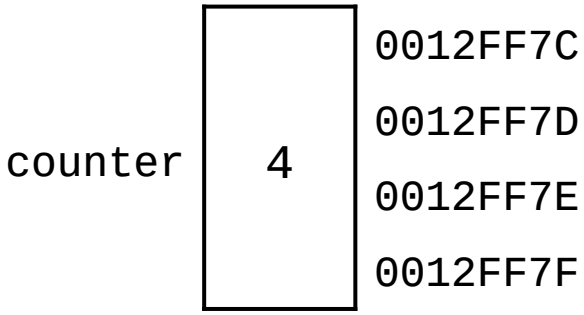


Output

```
1 2 3
```

Do While Statements

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```



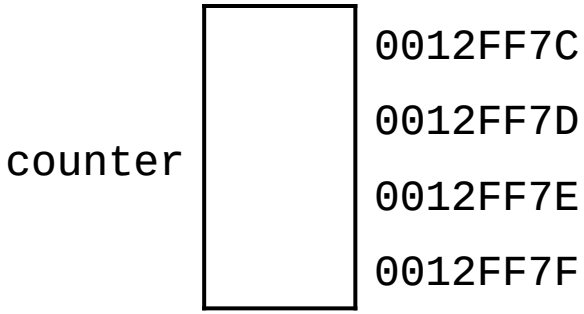
Output

```
1 2 3
```

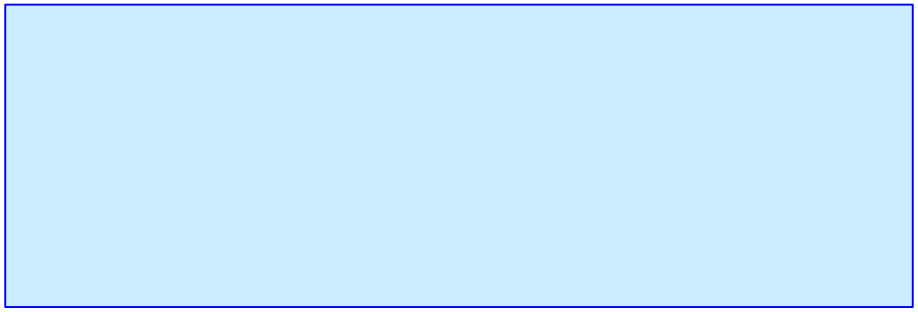
```
#include <iostream>
using namespace std;

int main()
{
   int counter{ 1 };

   do
   {
      cout << counter << ' ';
      ++counter;
   } while( counter <= 10 );

   cout << endl;
}
```

For Statements

```
for( int counter{ 4 }; counter <= 3; ++counter )
    cout << counter << ' ';
```

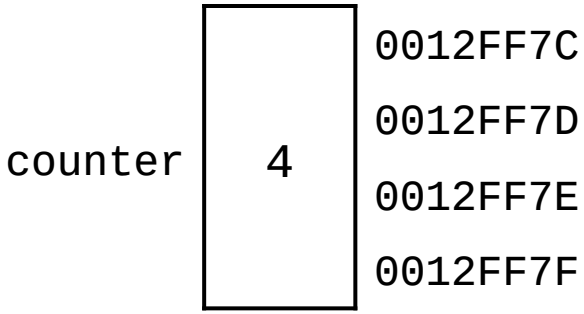


Output

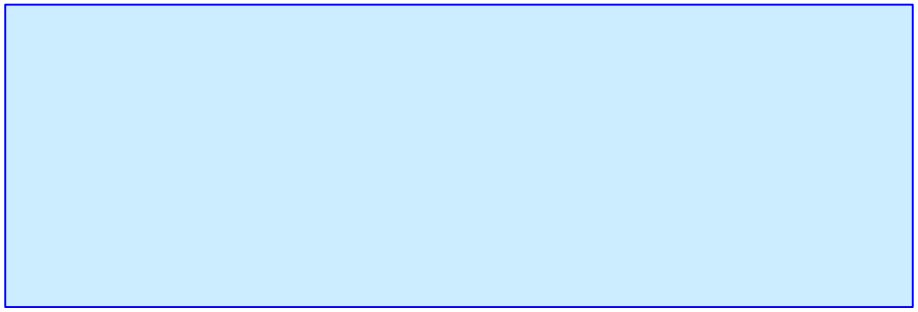


For Statements

```
for( int counter{ 4 }; counter <= 3; ++counter )
    cout << counter << ' ';
```



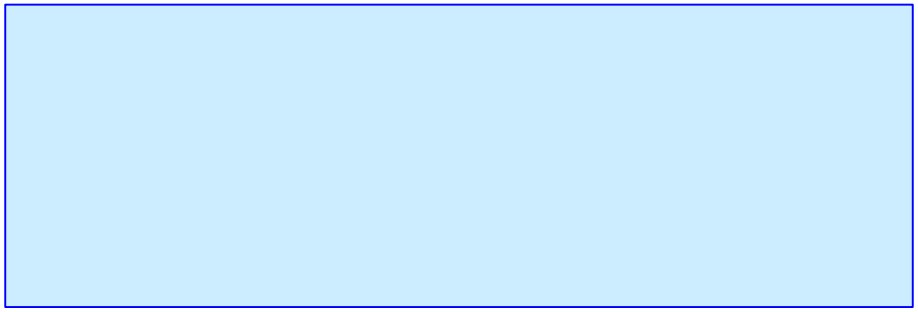
Output



While Statements

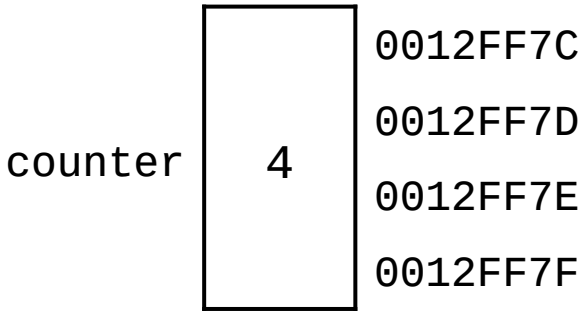
```
int counter{ 4 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Output



While Statements

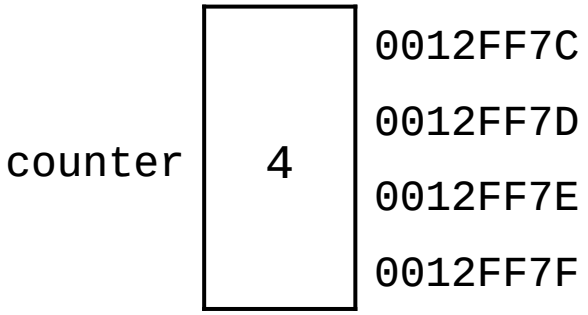
```
int counter{ 4 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```



Output

While Statements

```
int counter{ 4 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

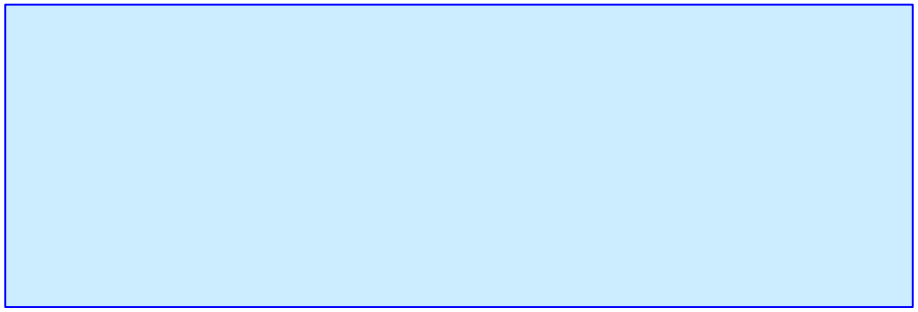


Output

Do While Statements

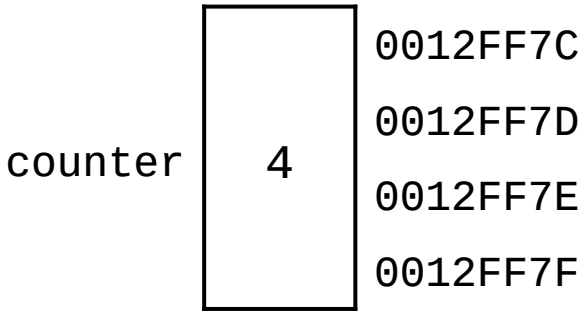
```
int counter{ 4 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

Output



Do While Statements

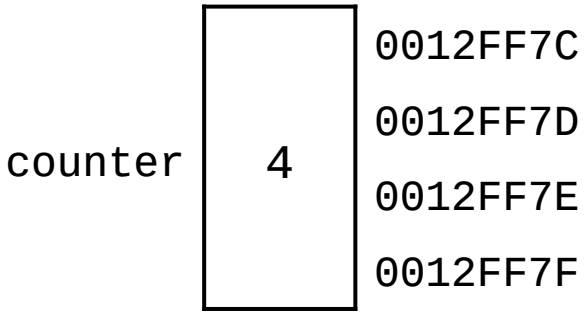
```
int counter{ 4 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```



Output

Do While Statements

```
int counter{ 4 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

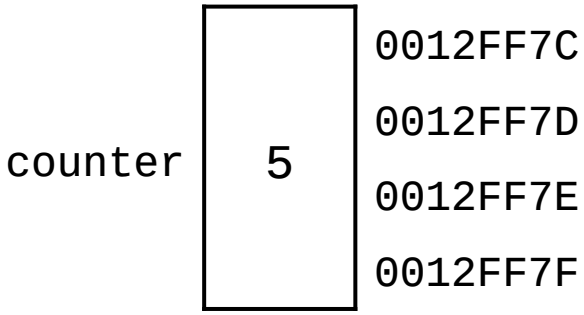


Output

```
4
```

Do While Statements

```
int counter{ 4 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

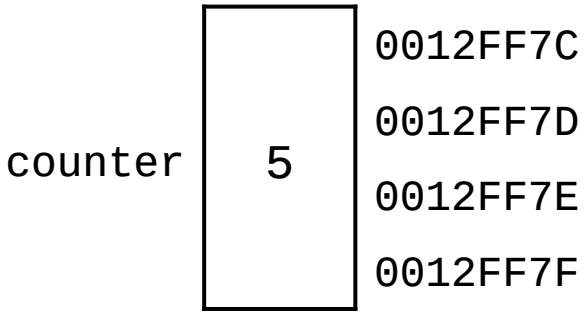


Output

```
4
```

Do While Statements

```
int counter{ 4 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```



Output

```
4
```

Comparison

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

```
int counter{ 1 };
do {
   cout << counter << ' ';
   ++counter;
} while( counter <= 3 );
cout << endl;
```

`for` Repetition Statement

- The for statement looks pretty simple in abstract:

```
for( init-statement; condition; end-expression )
   statement;
```

- It is possible to write for loops that omit any or all of the statements or expressions.
- An omitted condition-expression in a for loop should be treated as true.
- One might omit the `init-statement` if the control variable is initialized earlier in the program.
- One might omit the `end-expression` if the increment is calculated by statements in the body of the `for` or if no increment is needed.

```
#include <iostream>
using namespace std;

int main()
{
   for( int counter{ 1 }; counter <= 10; ++counter )
      cout << counter << ' ';

   cout << endl;
}
```

`for` Repetition Statement

The three expressions in the `for` statement header are optional.

```
for( int counter{ 1 }; counter <= 3; ++counter )
    cout << counter << ' ';
cout << endl;
```

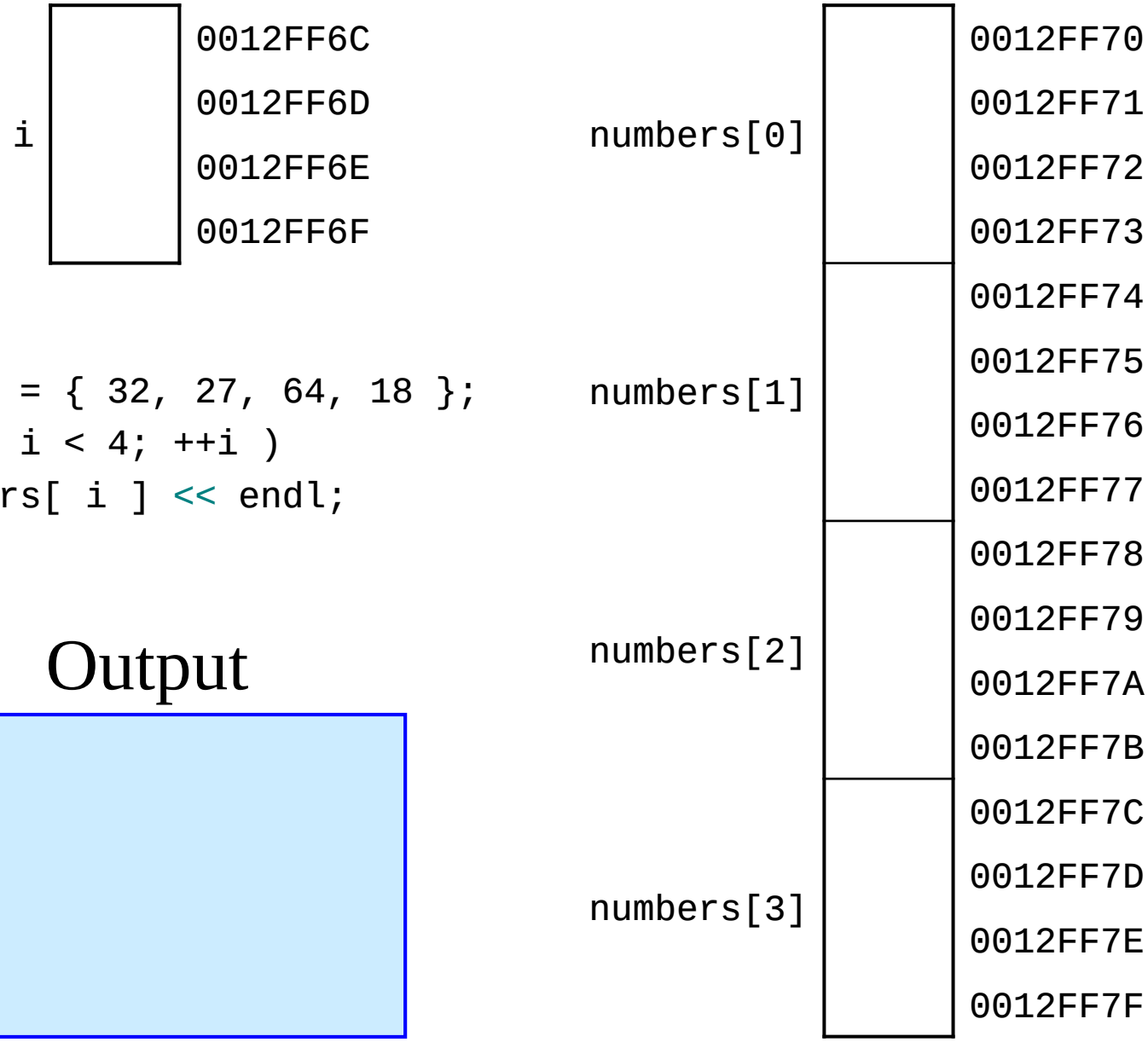
```
int counter{ 1 };
while( counter <= 3 )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

```
int counter{ 1 };
for( ; counter <= 3; )
{
   cout << counter << ' ';
   ++counter;
}
cout << endl;
```

Examples Using the for Statement

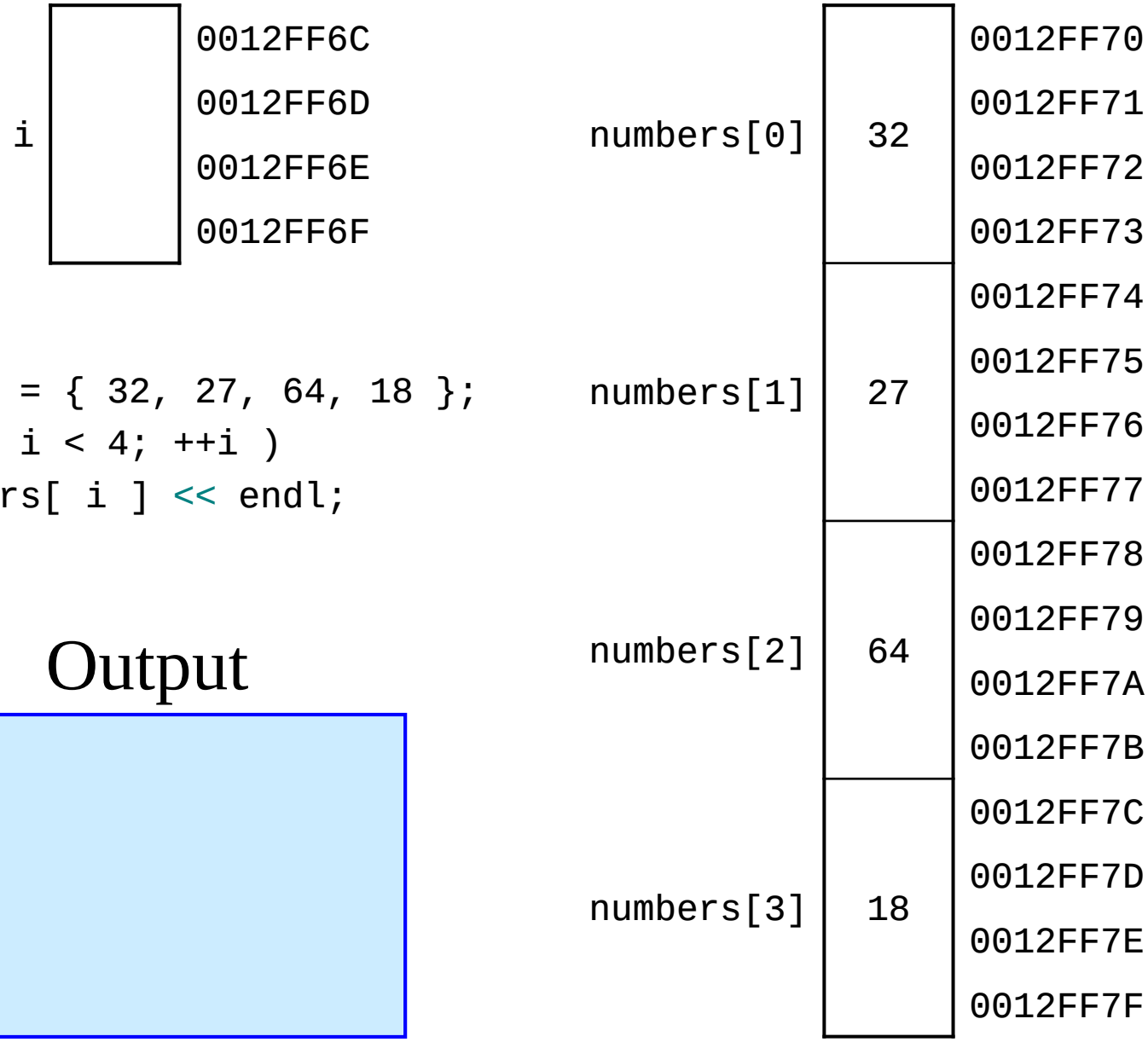
```
for( int i{ 1 }; i <= 100; i++ )

for( int i = { 100 }; i >= 1; i-- )

for( int i = { 7 }; i <= 77; i += 7 )

for( int i = { 20 }; i >= 2; i -= 2 )

for( int i = { 2 }; i <= 20; i += 3 )

for( int i = { 99 }; i >= 0; i -= 11 )
```

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int numbers[ 4 ] = { 32, 27, 64, 18 };
   cout << "Index" << setw( 7 ) << "Value" << endl;
   for( int i{ 0 }; i < 4; ++i )
      cout << setw( 5 ) << i << setw( 7 ) << numbers[ i ] << endl;
}
```



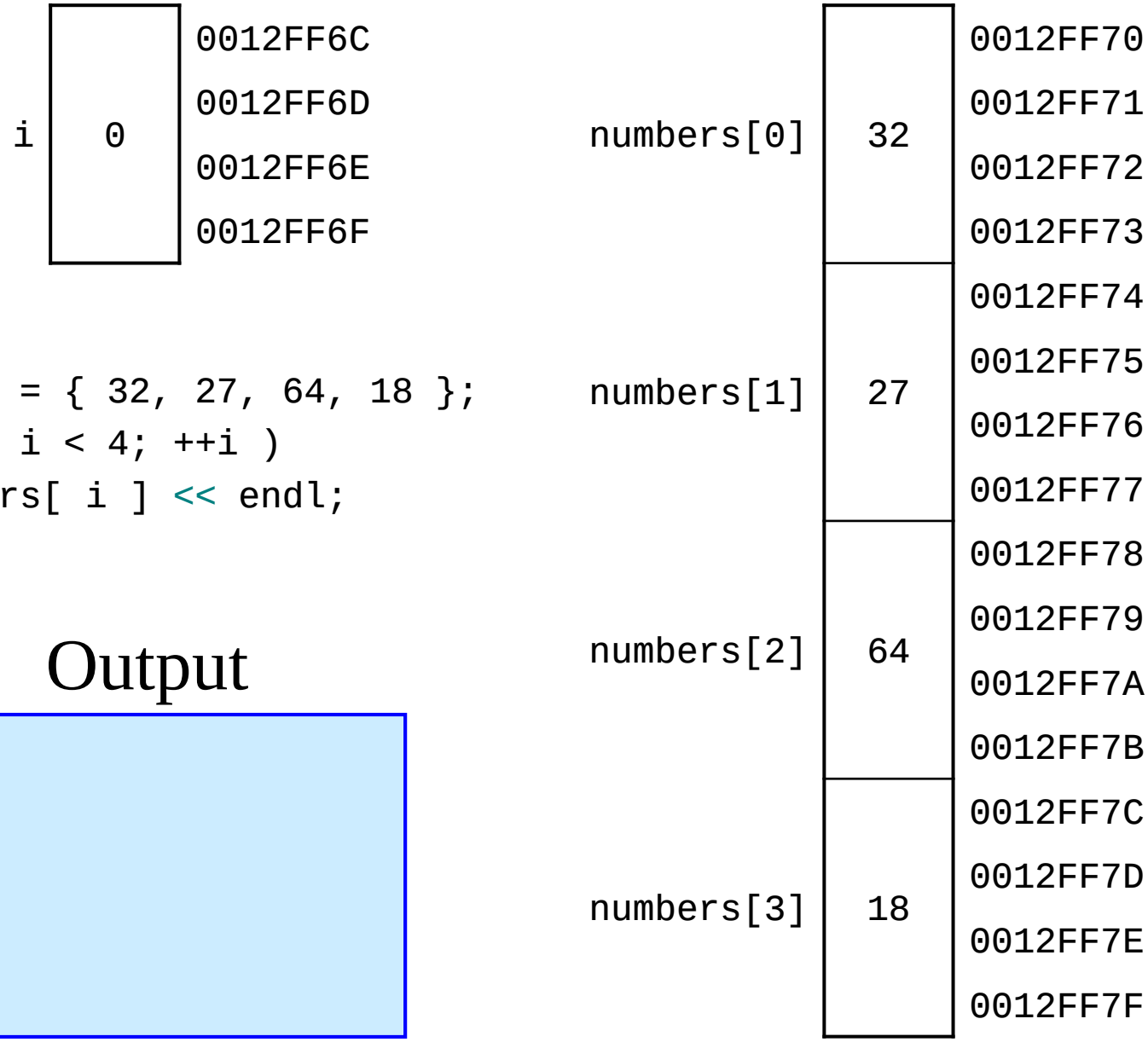

```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output



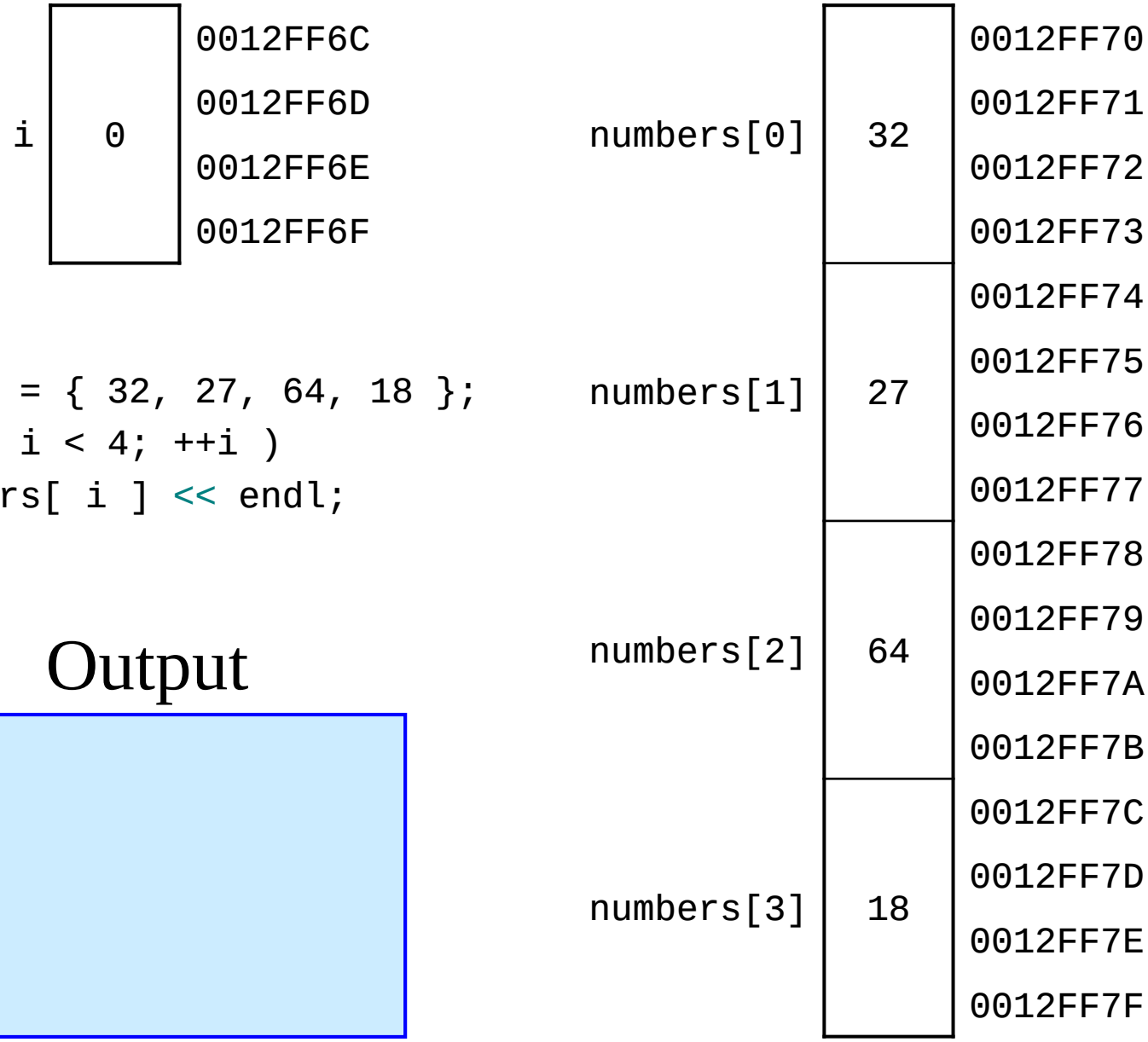

```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

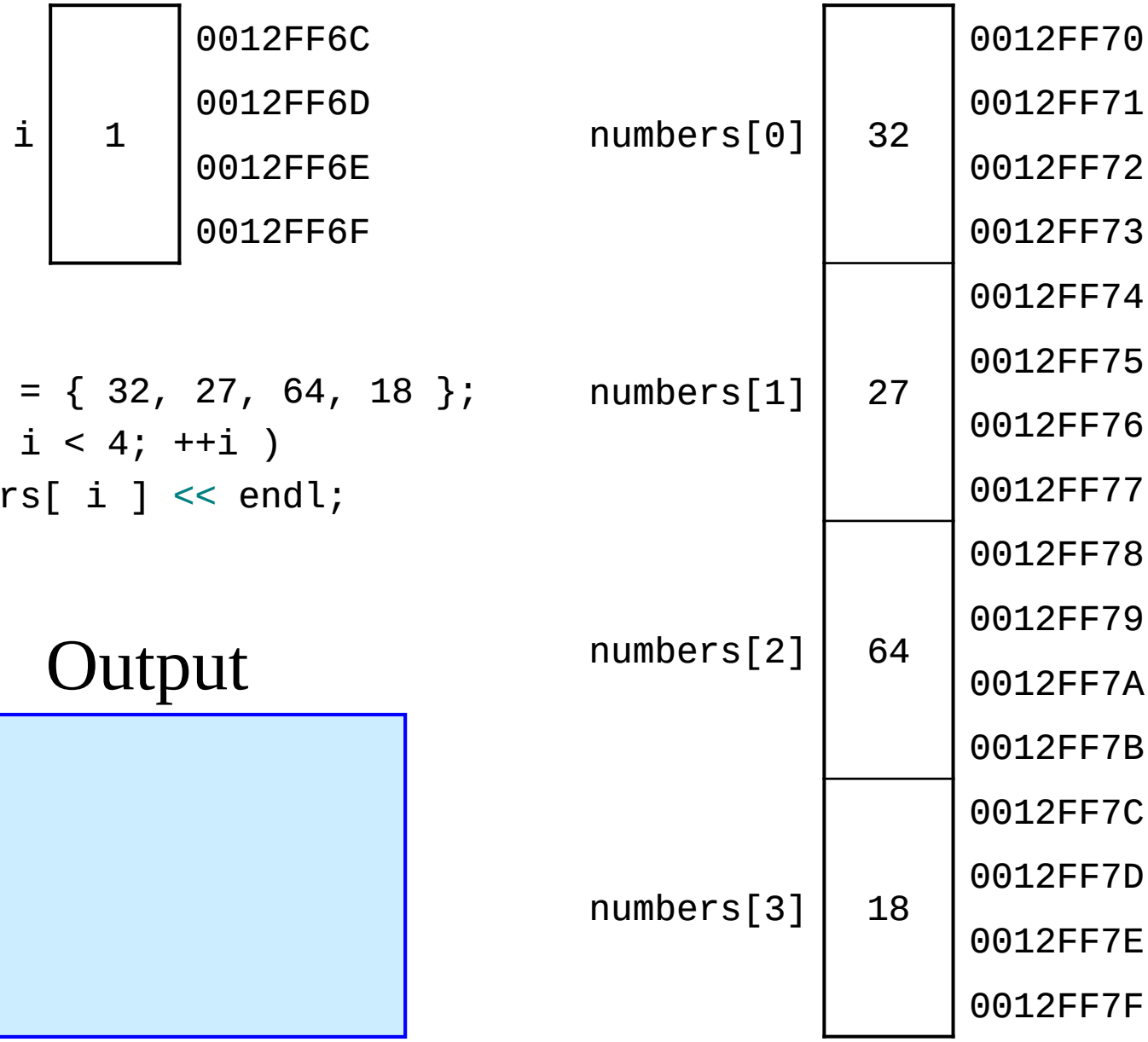
Output




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

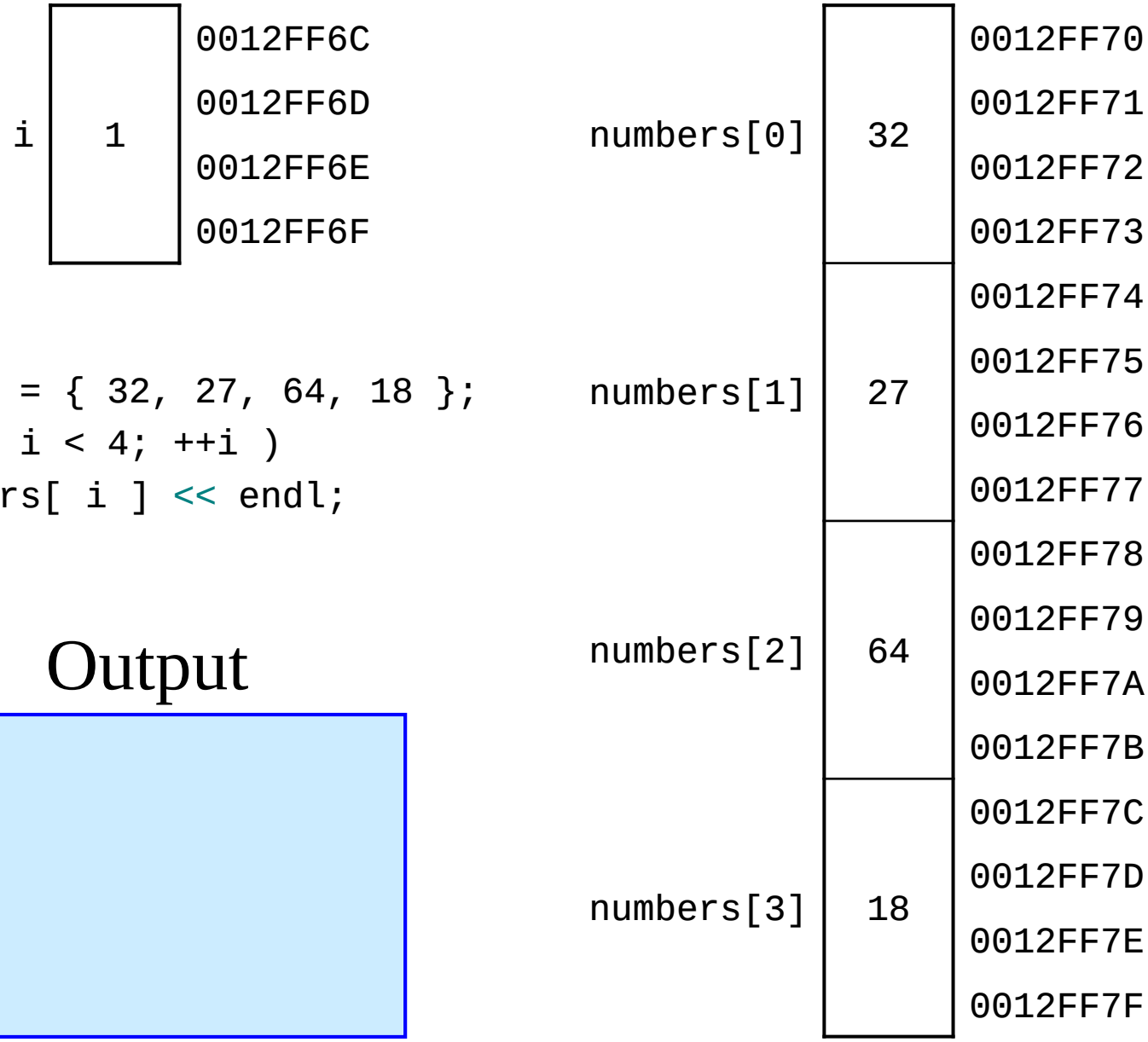
```
32
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

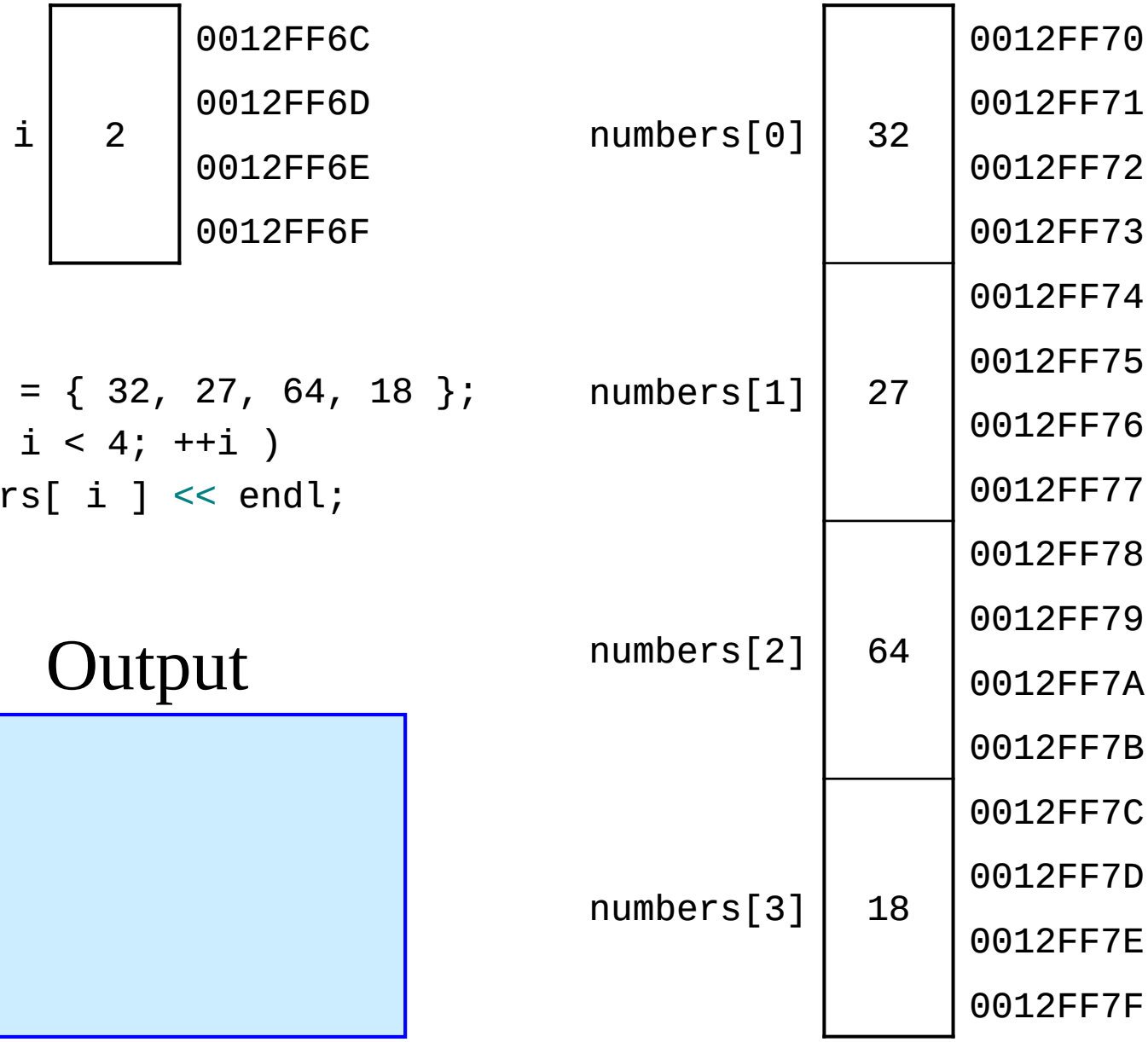
```
32
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

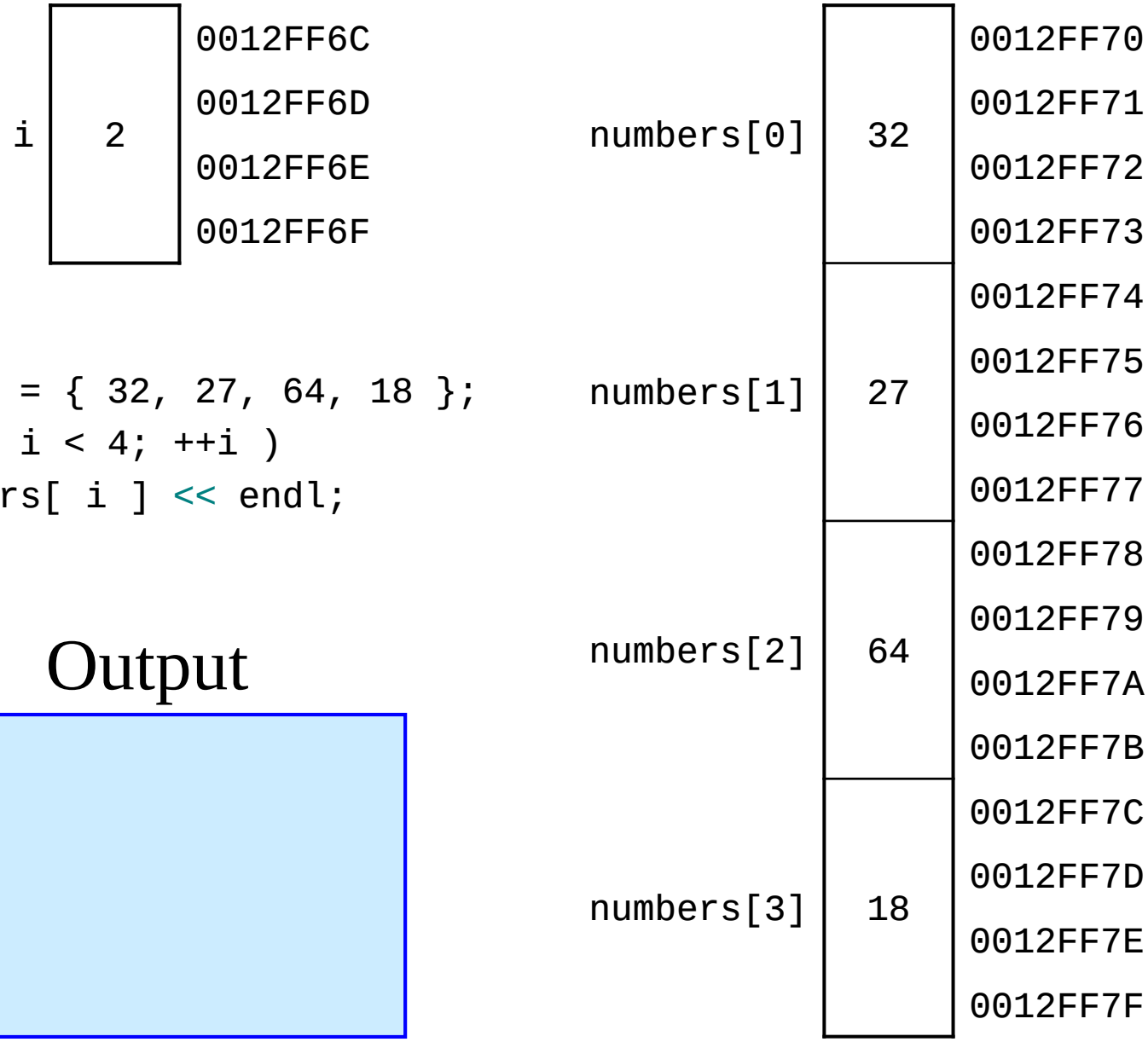
```
32
27
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

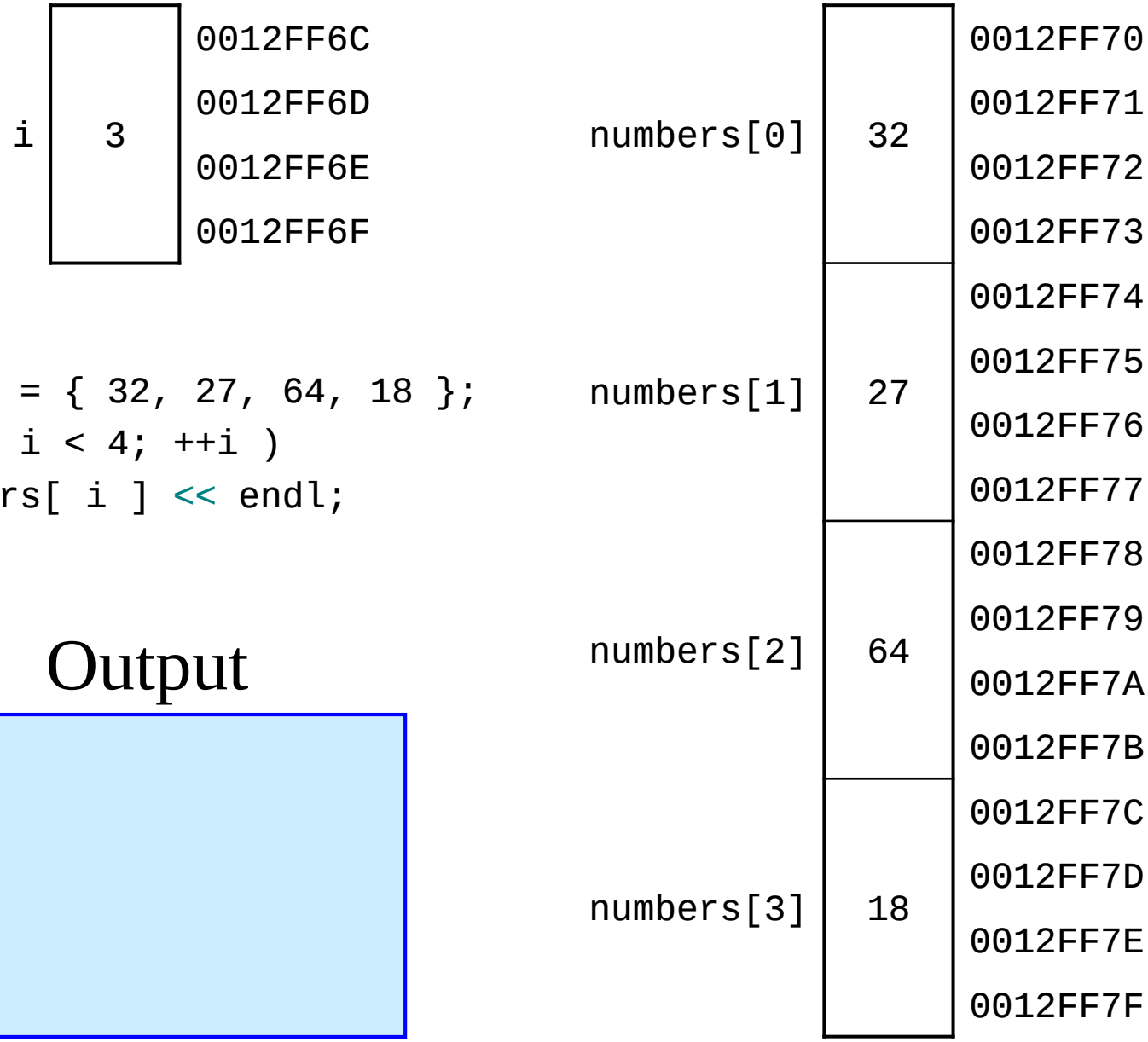
```
32
27
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

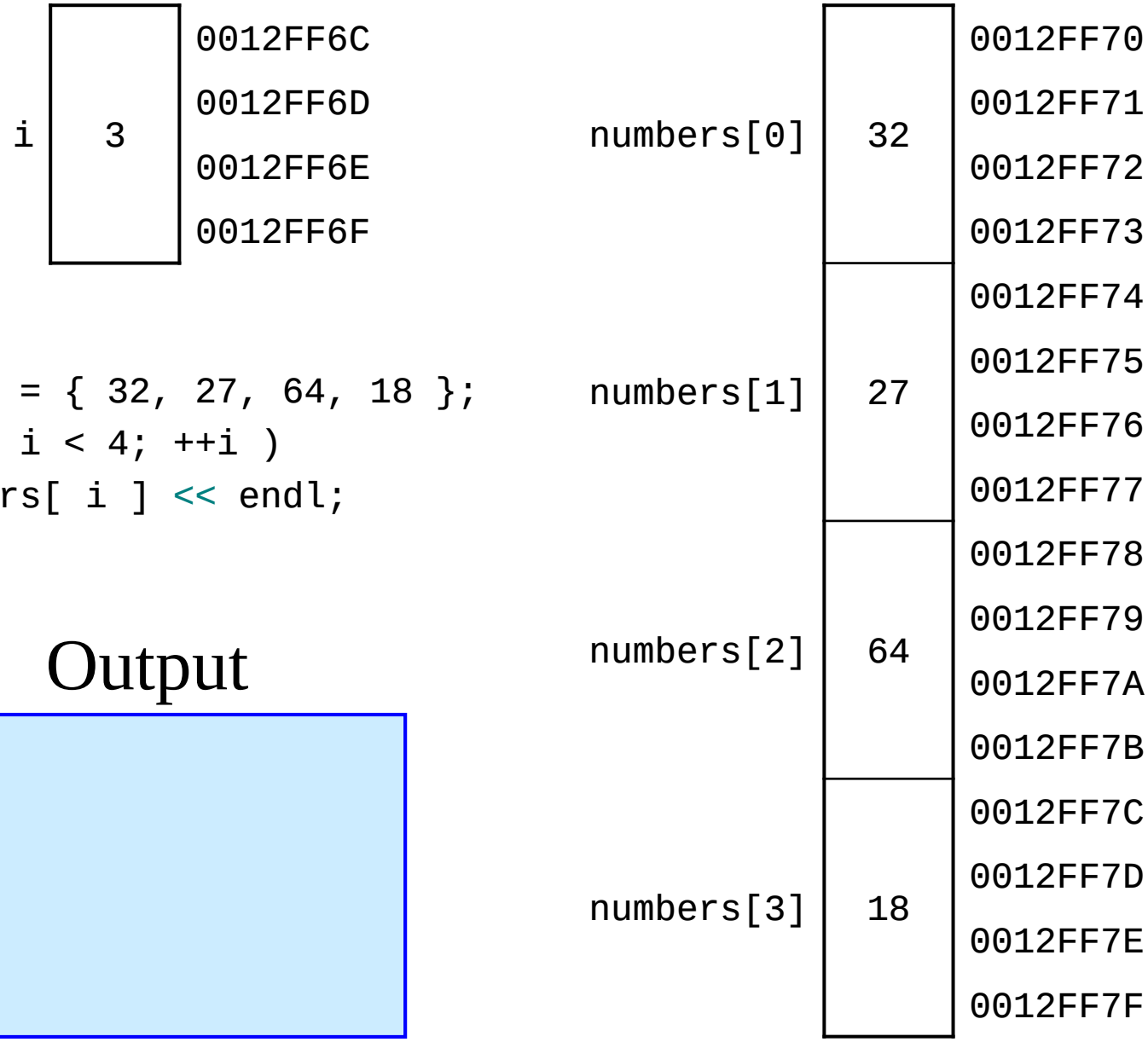
```
32
27
64
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

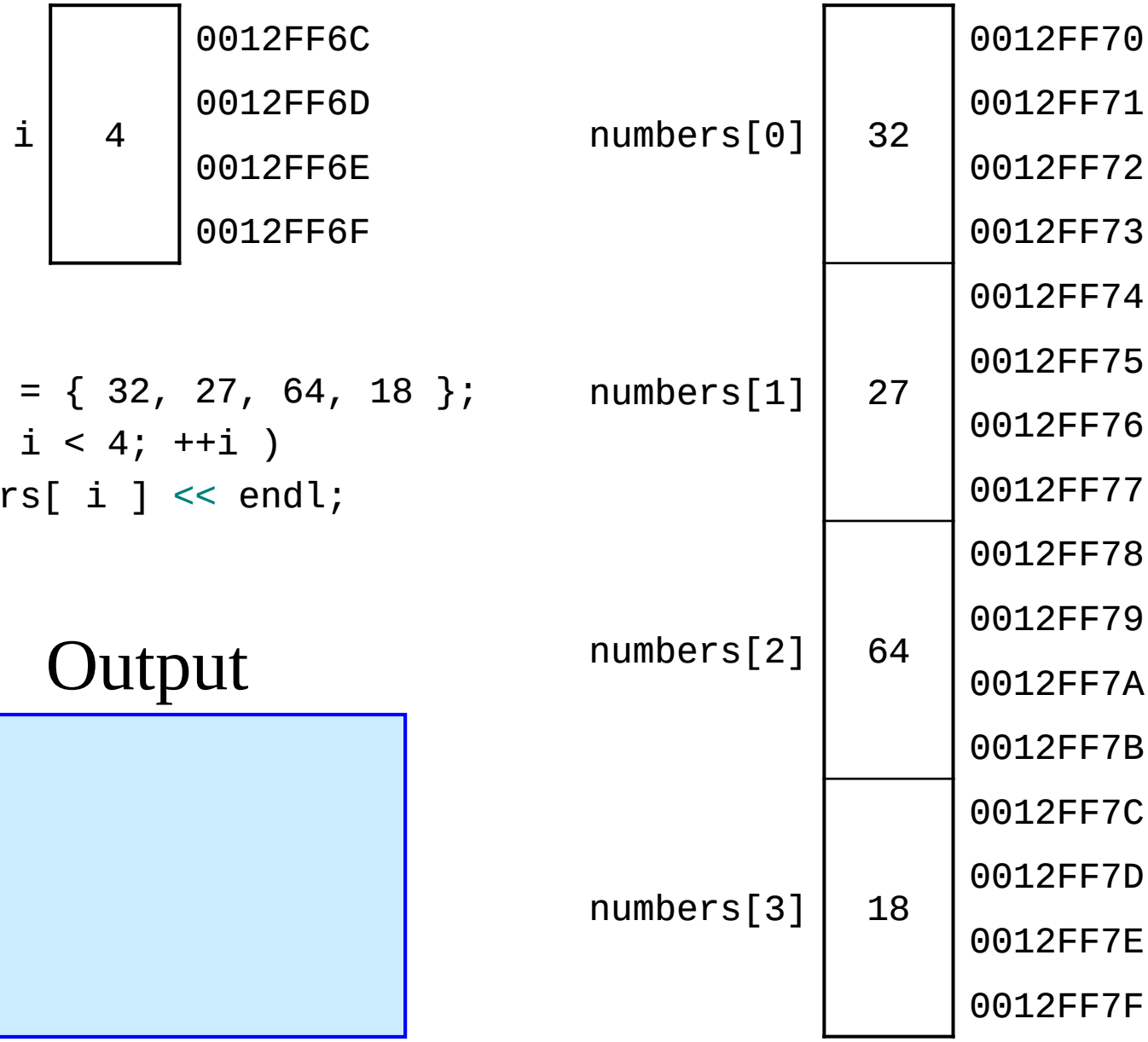
```
32
27
64
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

```
32
27
64
18
```




```
int numbers[ 4 ] = { 32, 27, 64, 18 };
for( int i{ 0 }; i < 4; ++i )
   cout << numbers[ i ] << endl;
```

Output

```
32
27
64
18
```

```
Index  Value
    0     32
    1     27
    2     64
    3     18
```

const variable

```
#include <iostream>
using namespace std;

int main()
{
   const int number{ 10 };
   cout << number << endl;
}
```

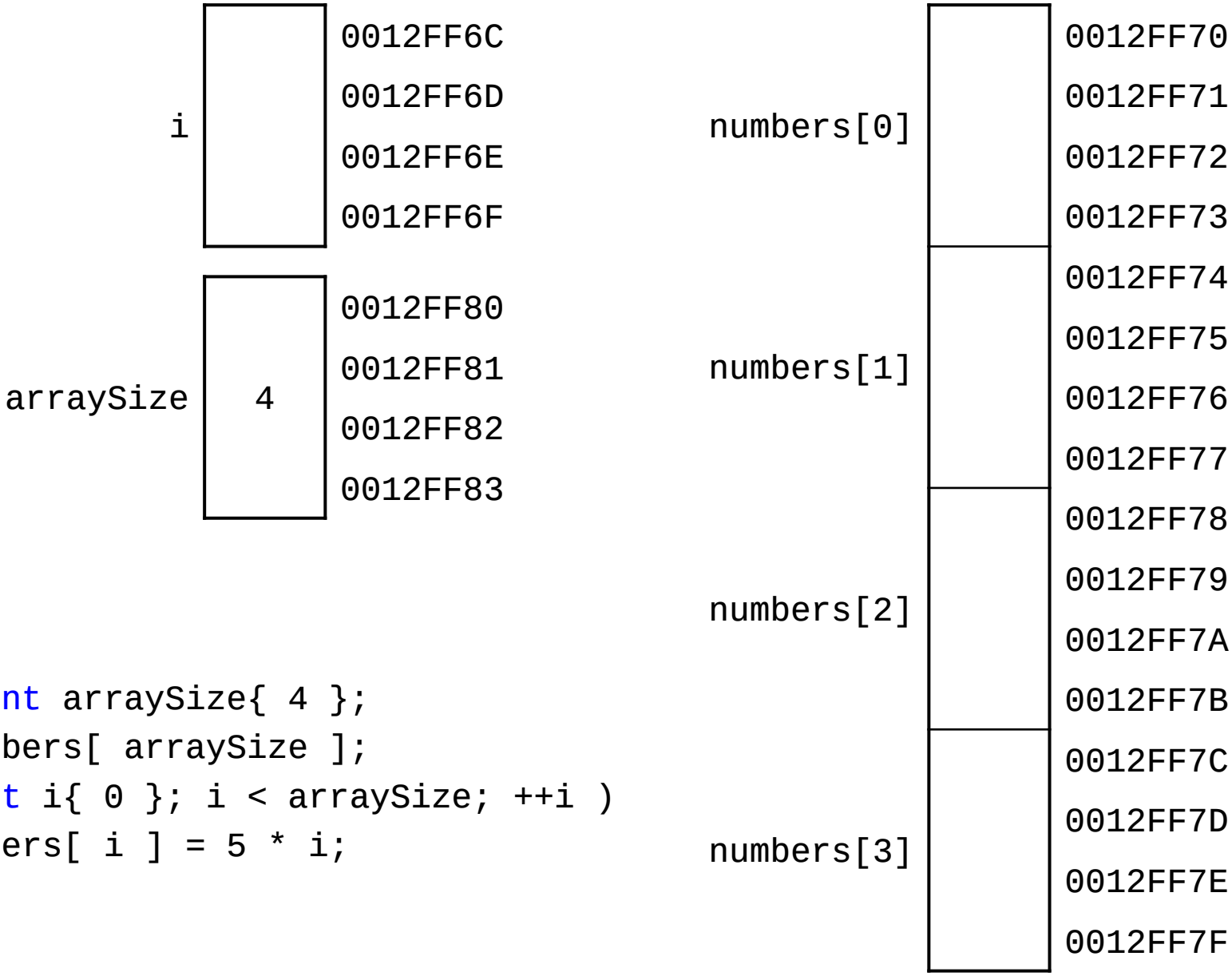
const variable

```
#include <iostream>
using namespace std;

int main()
{
   const int number; // Error: number must be initialized
}
```

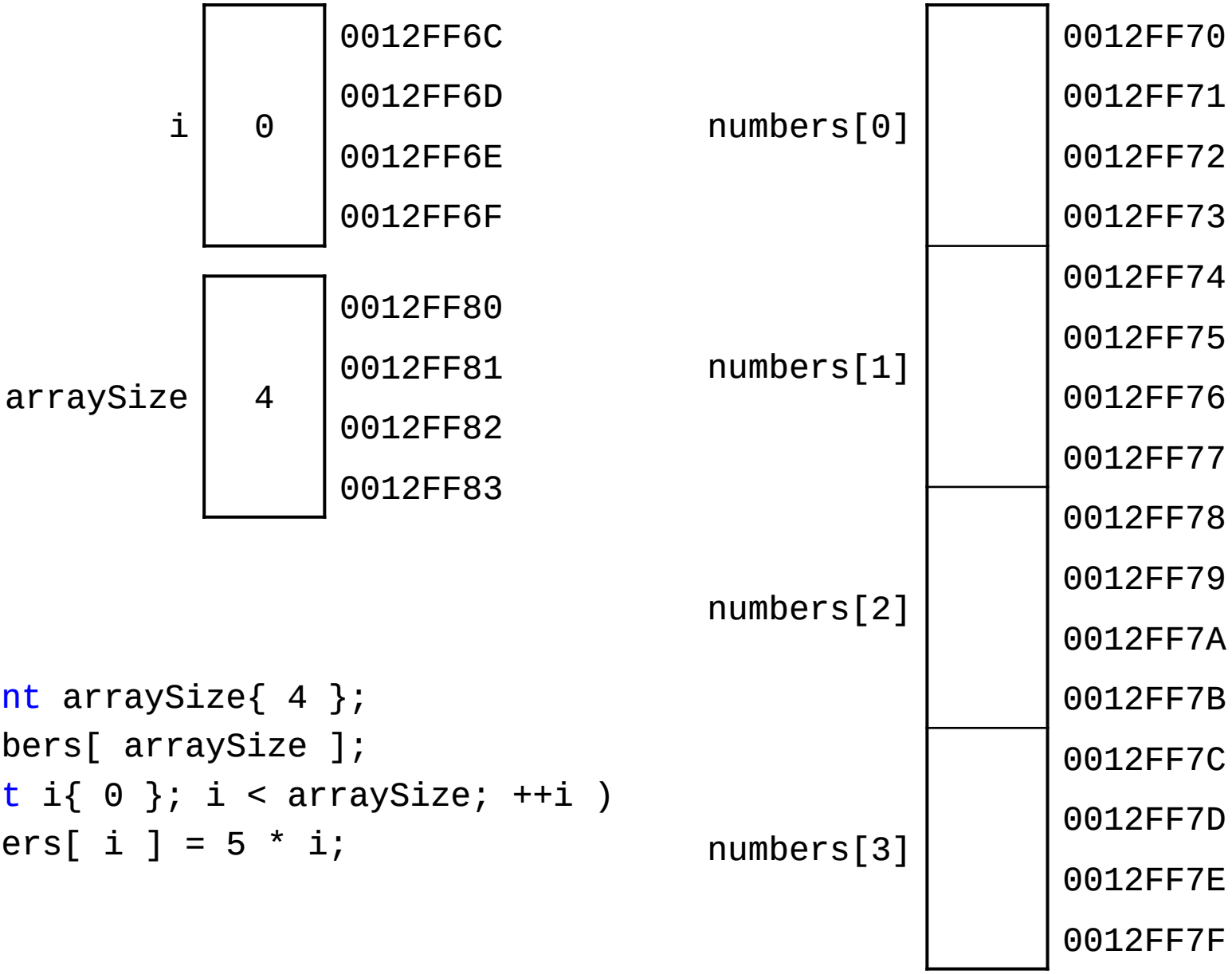
const variable

```
#include <iostream>
using namespace std;

int main()
{
   const int number{ 10 };
   number = 20; // Error: cannot modify a const variable
}
```

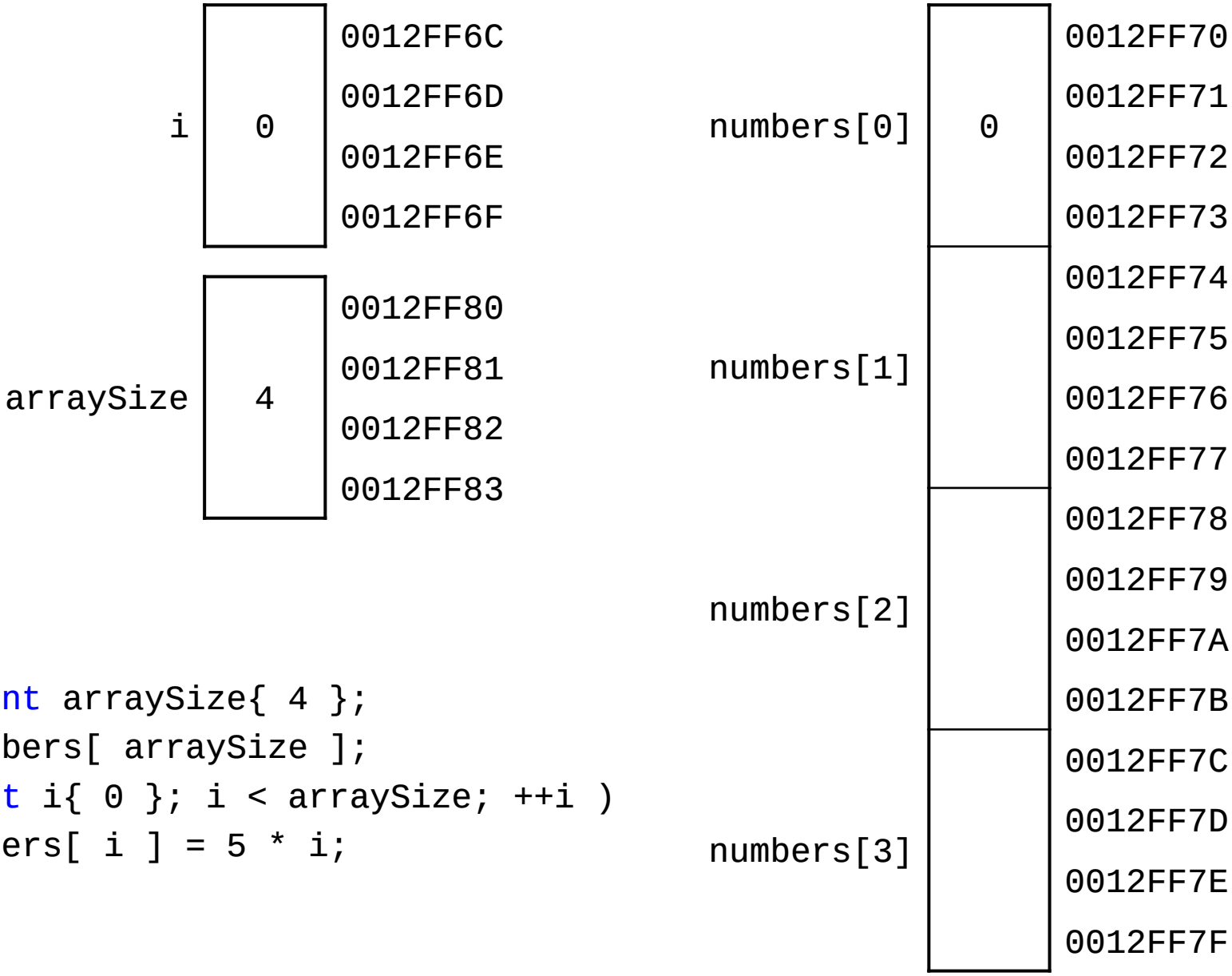
```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const int arraySize{ 4 };
   int numbers[ arraySize ];
   for( int i{ 0 }; i < arraySize; ++i )
      numbers[ i ] = 5 * i;

   cout << "Index" << setw( 7 ) << "Value" << endl;
   for( int i{ 0 }; i < arraySize; ++i )
      cout << setw( 5 ) << i << setw( 7 ) << numbers[ i ] << endl;
}
```



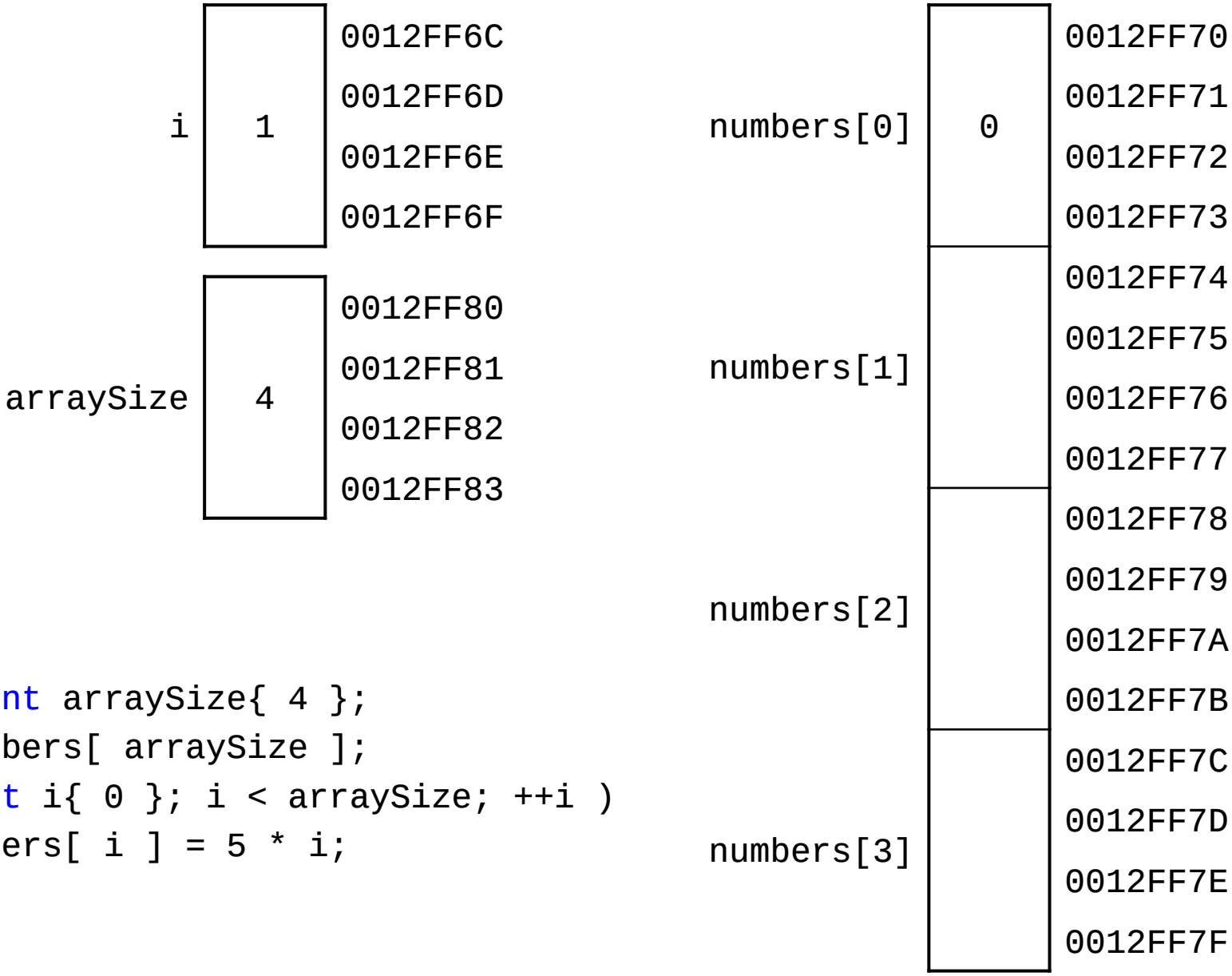

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```



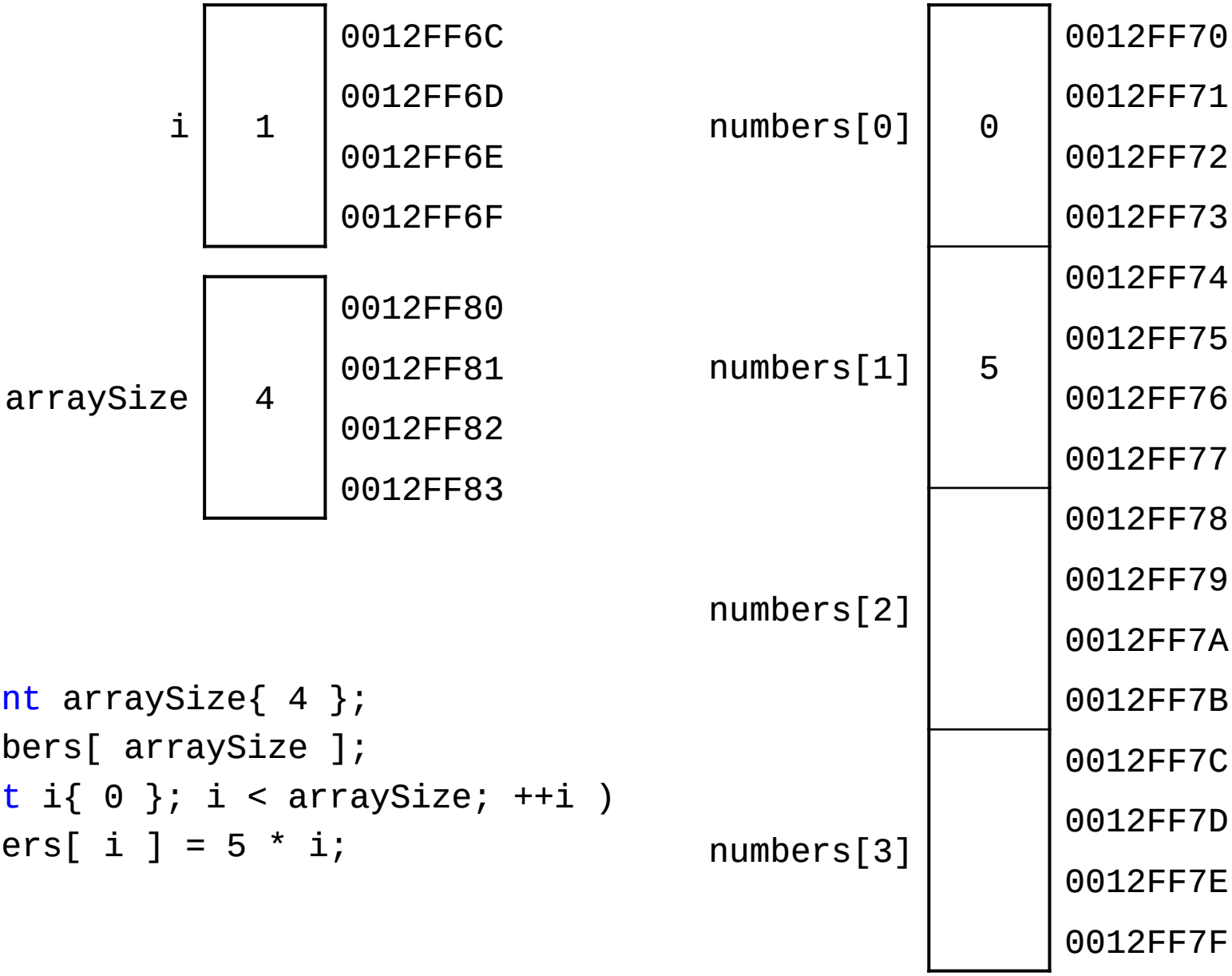

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```



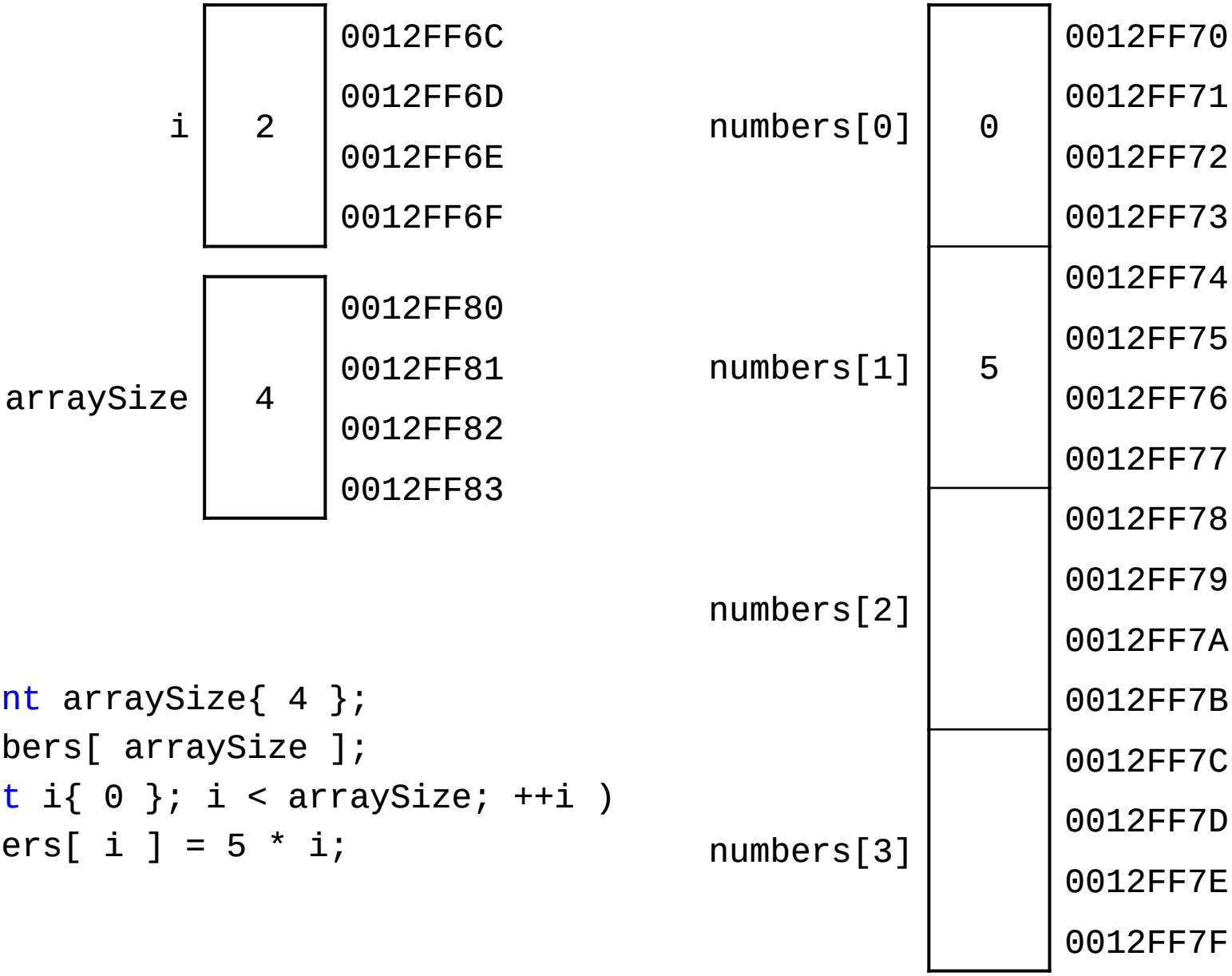

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```



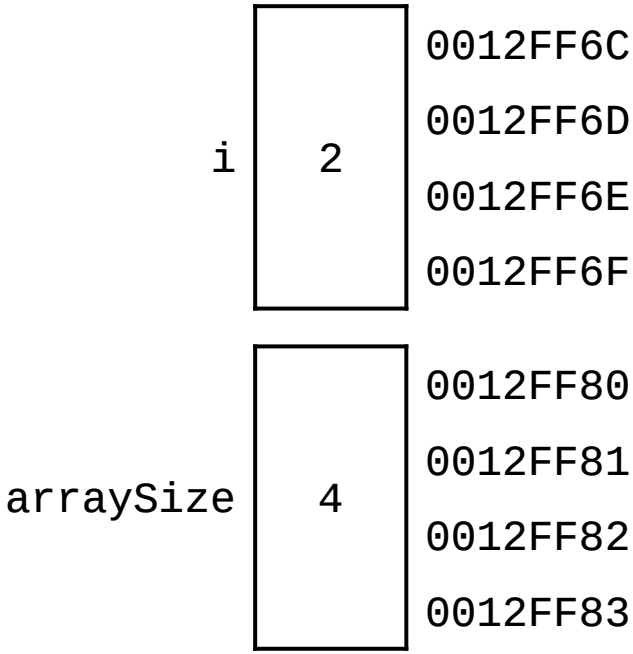

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```

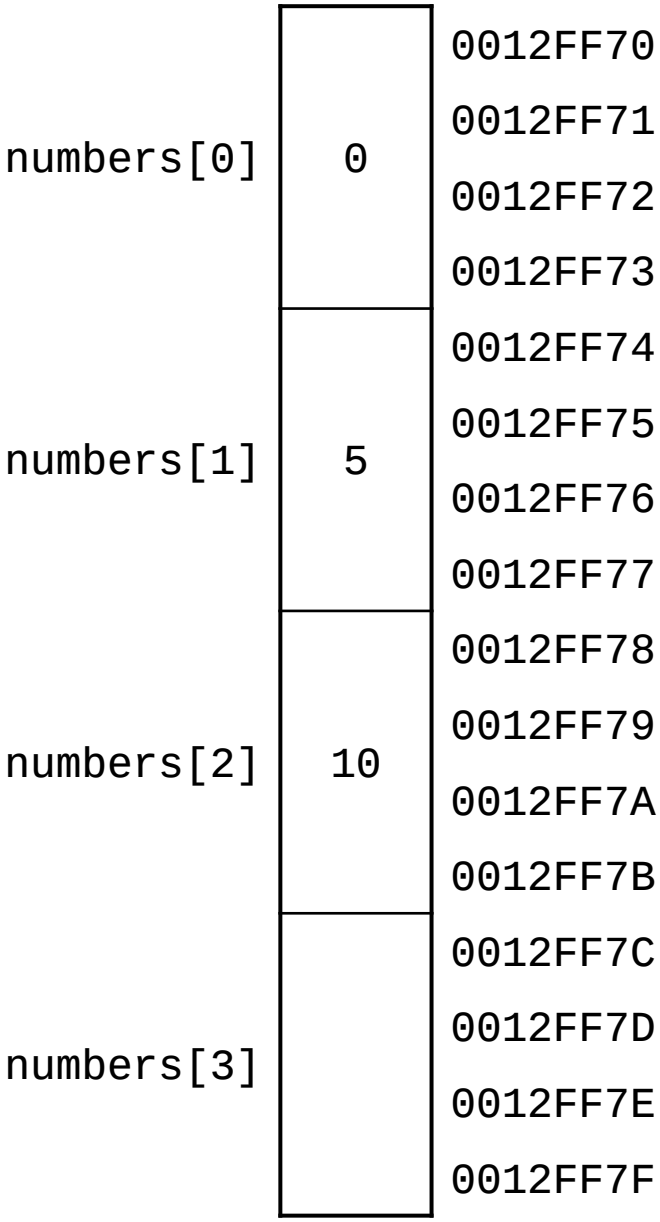



```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```

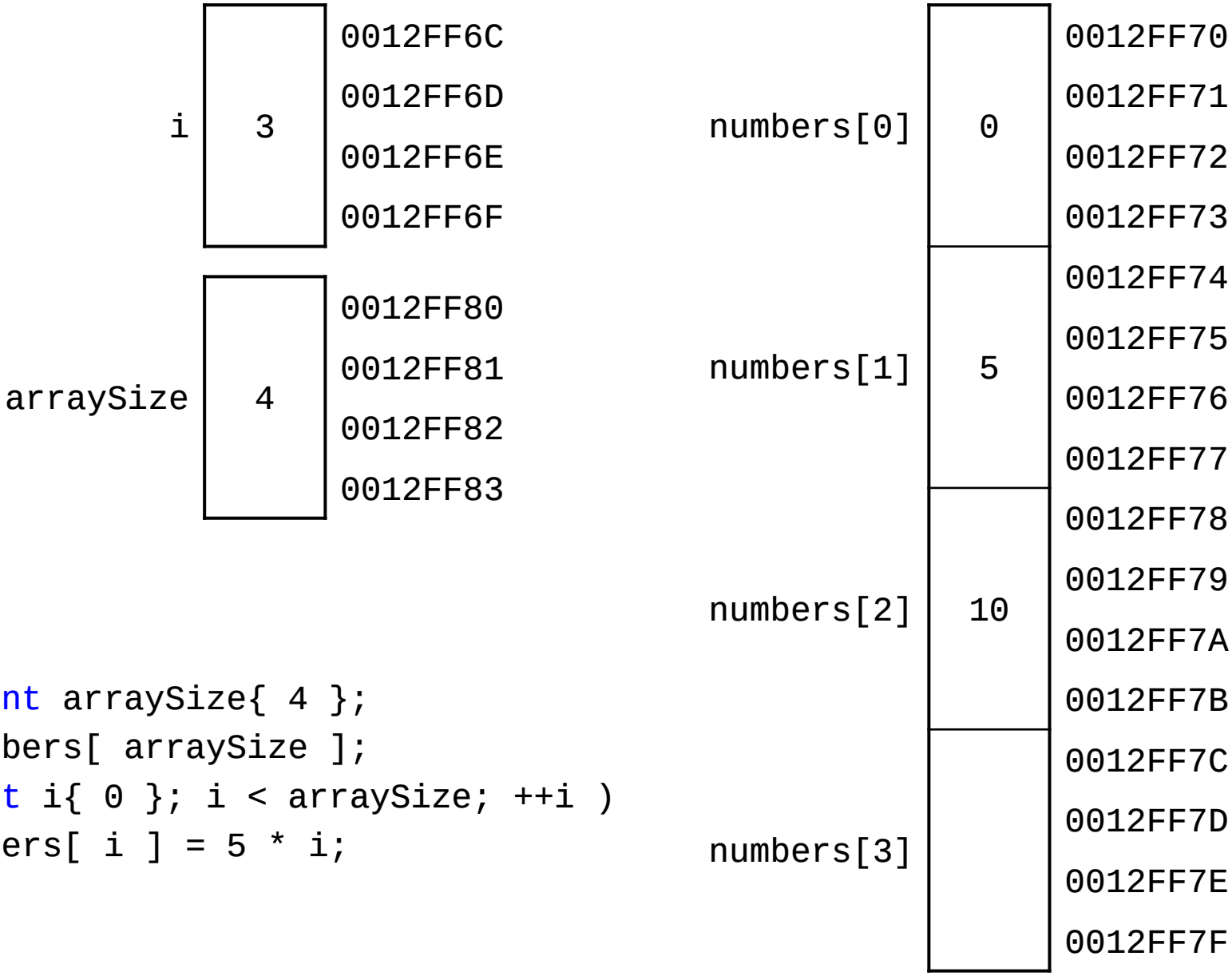



```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```

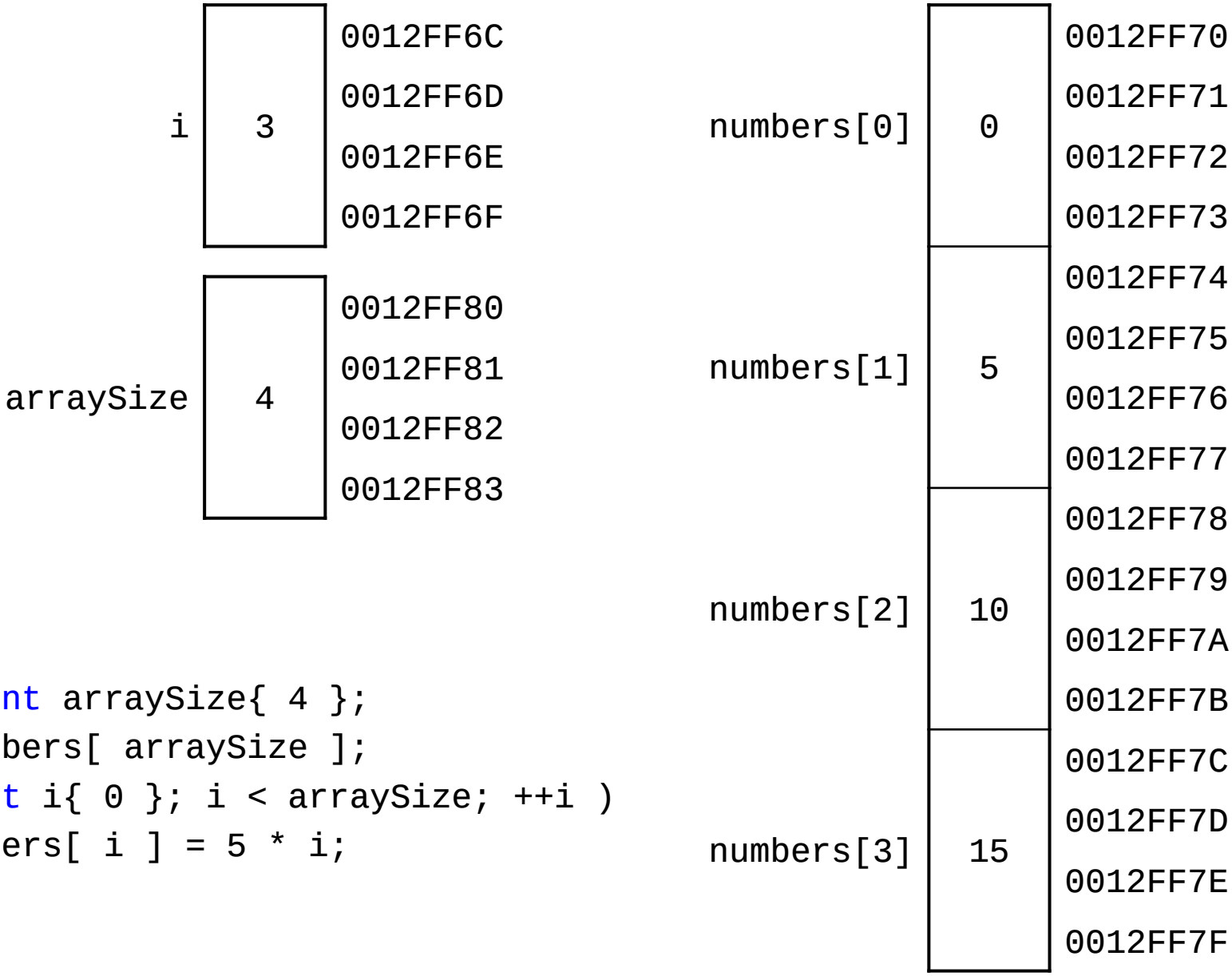




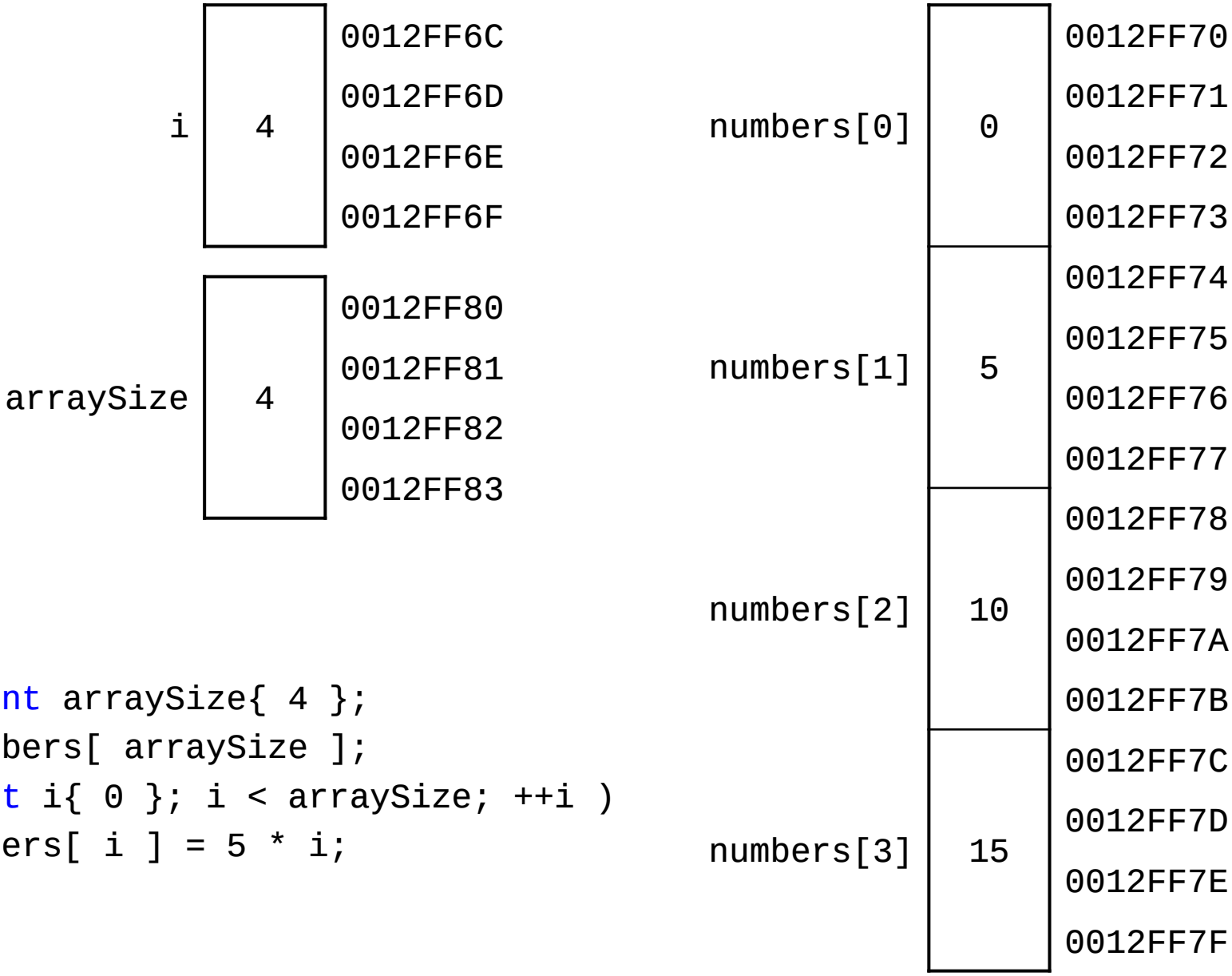

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```




```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```



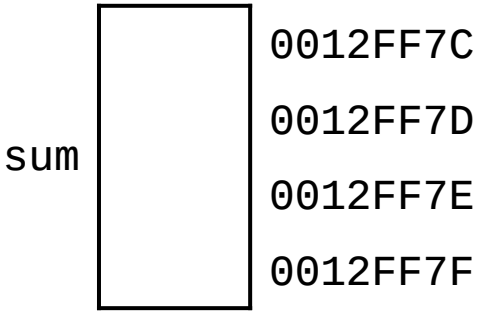

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```



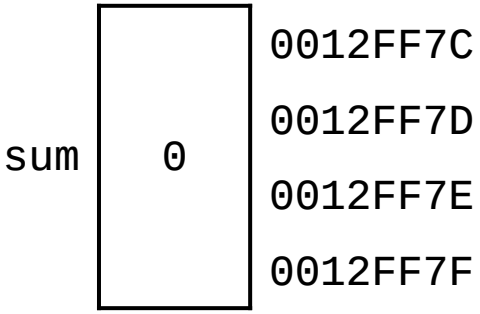

```
const int arraySize{ 4 };
int numbers[ arraySize ];
for( int i{ 0 }; i < arraySize; ++i )
   numbers[ i ] = 5 * i;
```

```
Index  Value
    0      0
    1      5
    2     10
    3     15
```

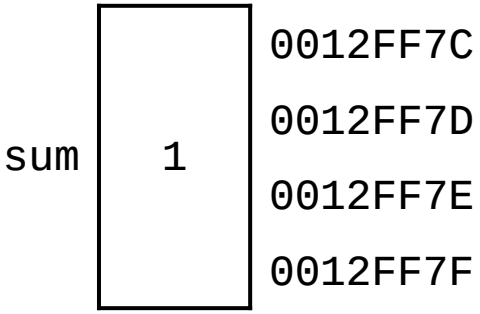
```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```



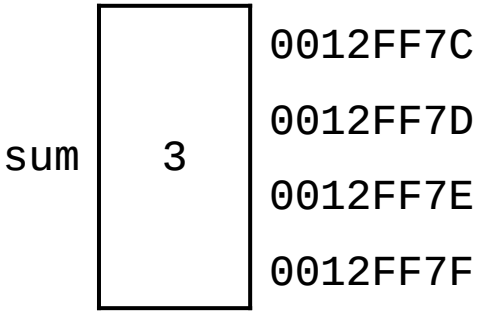
```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```



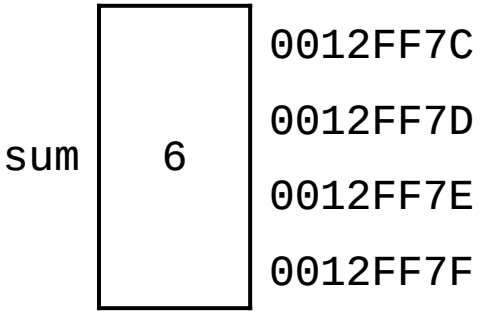
```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```



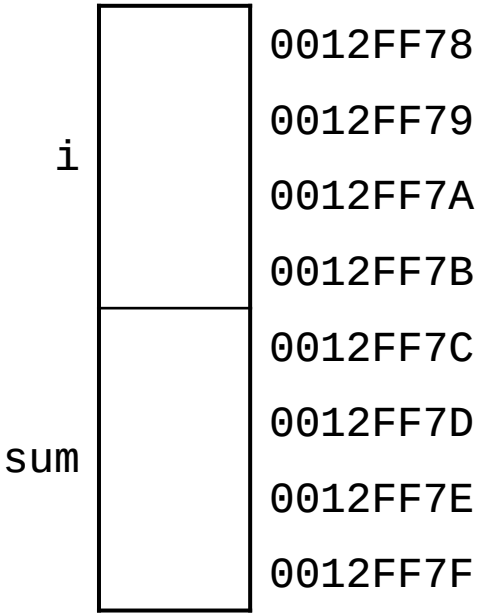
```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```



```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

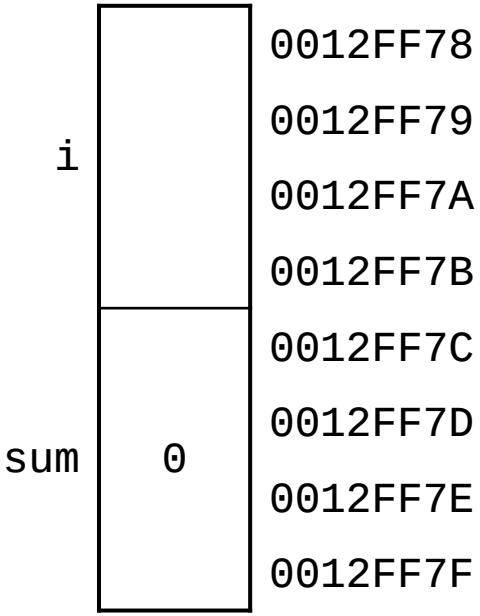


```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



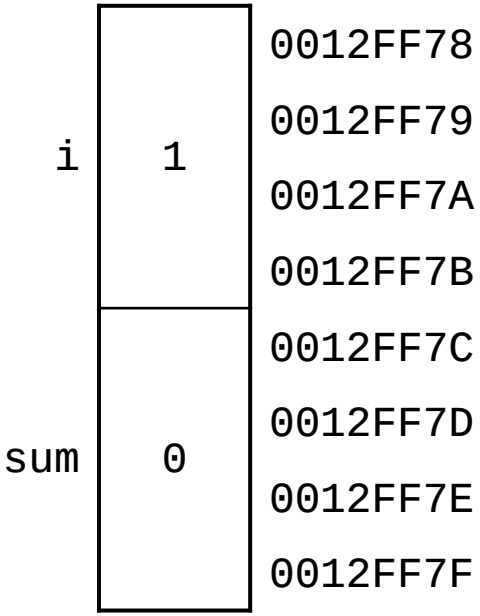

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



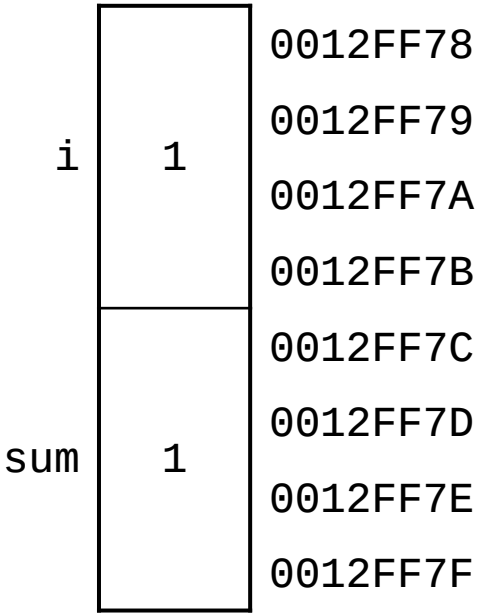

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



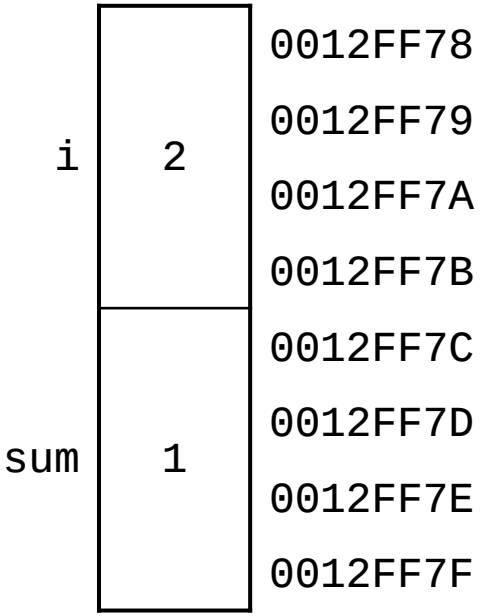

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



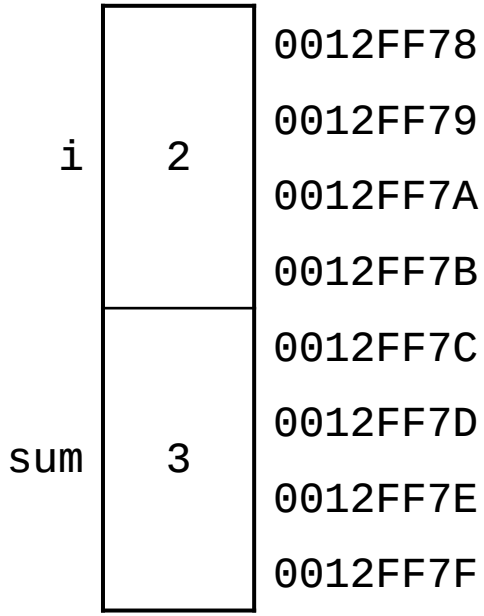

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



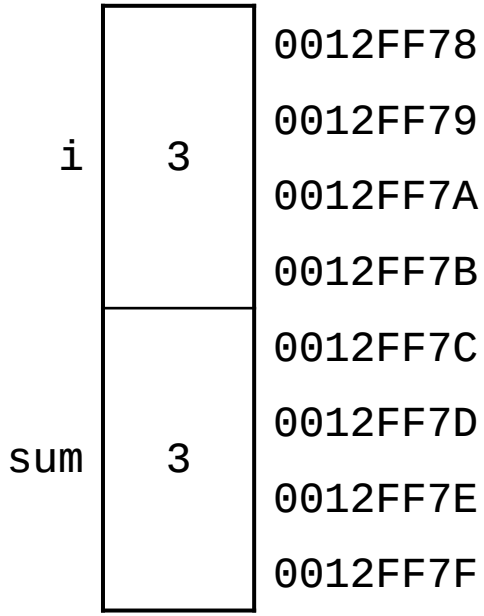

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



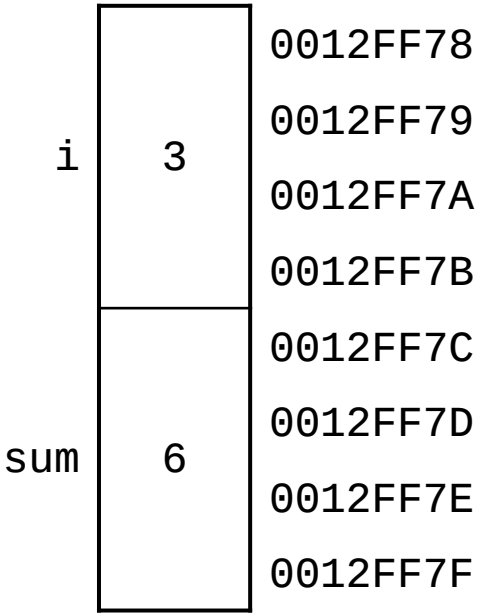

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



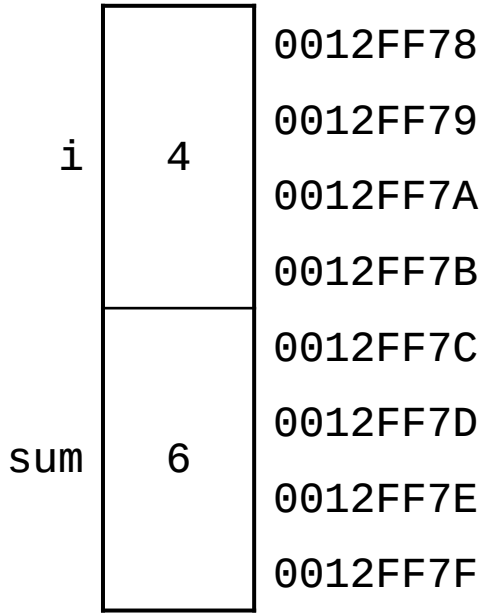

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



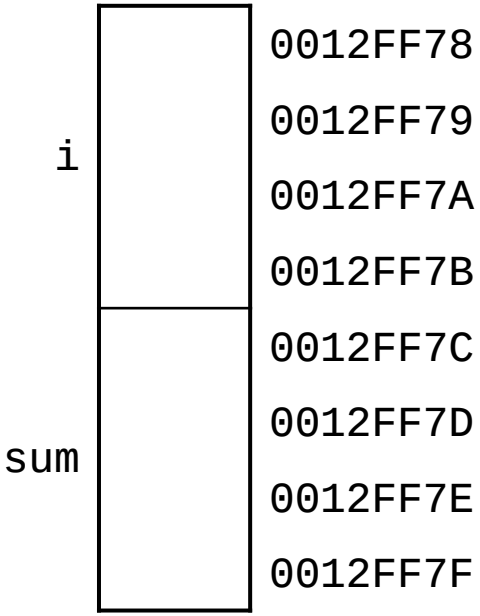

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for(            ;        ;      )
   sum += i;
```



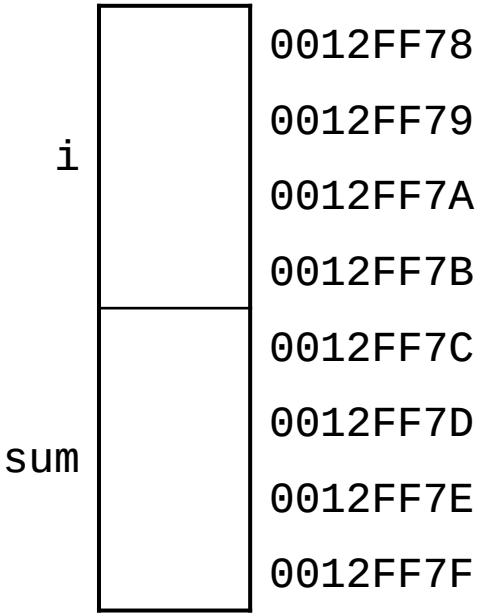

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 };        ;      )
   sum += i;
```



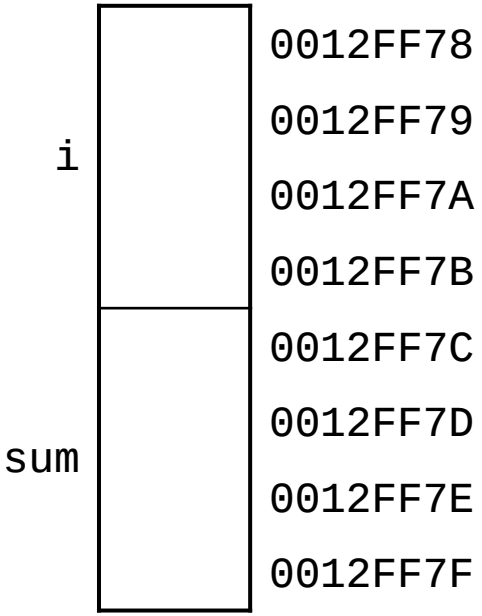

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 };        ; ++i )
   sum += i;
```



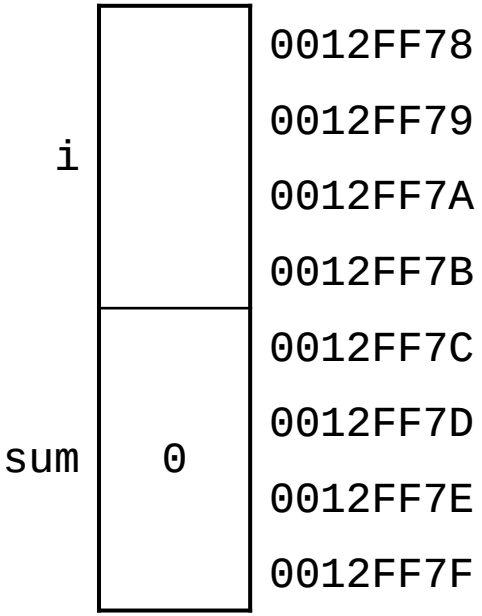

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



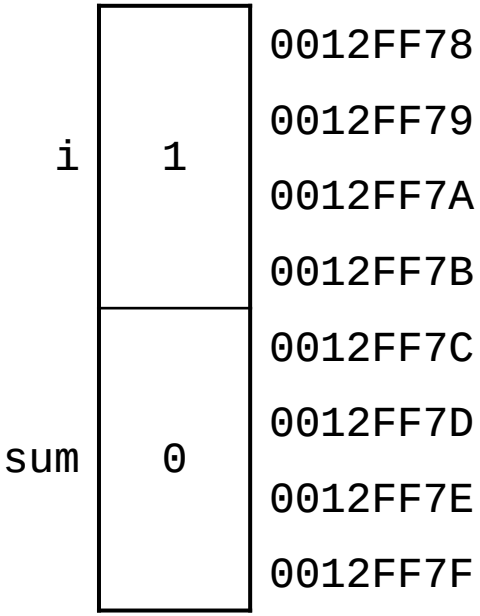

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



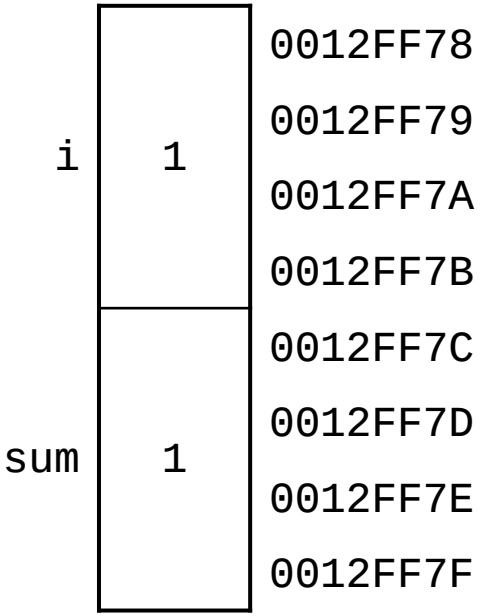

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



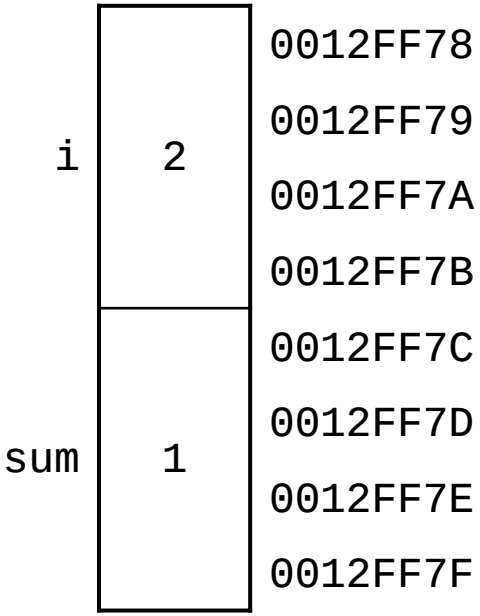

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



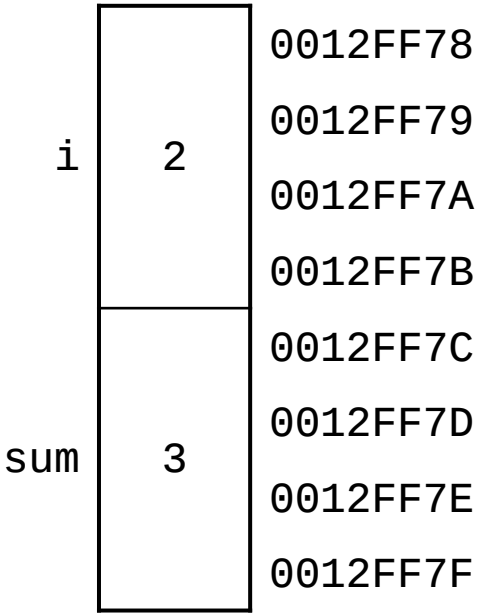

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



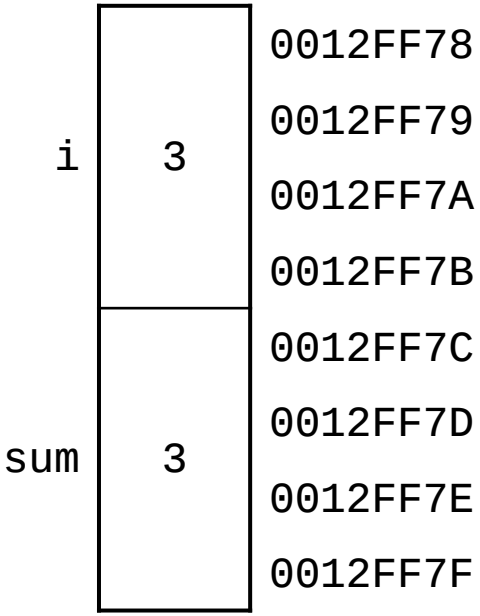

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



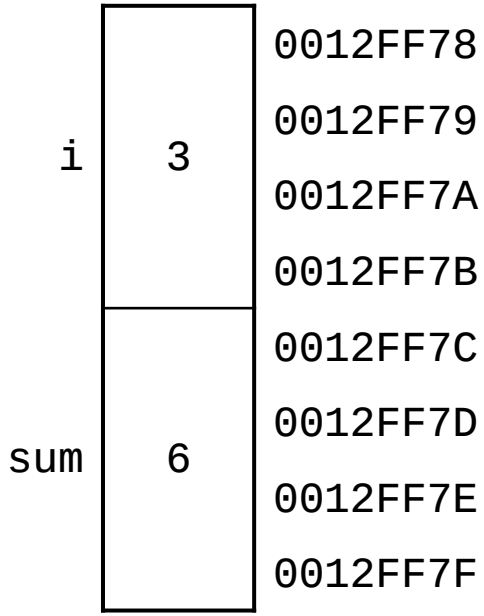

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```



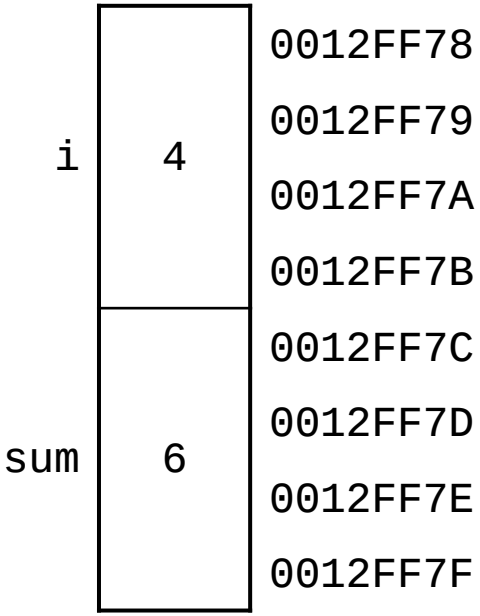

```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```




```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

```
int sum{ 0 };
for( int i{ 1 }; i <= 3; ++i )
   sum += i;
```

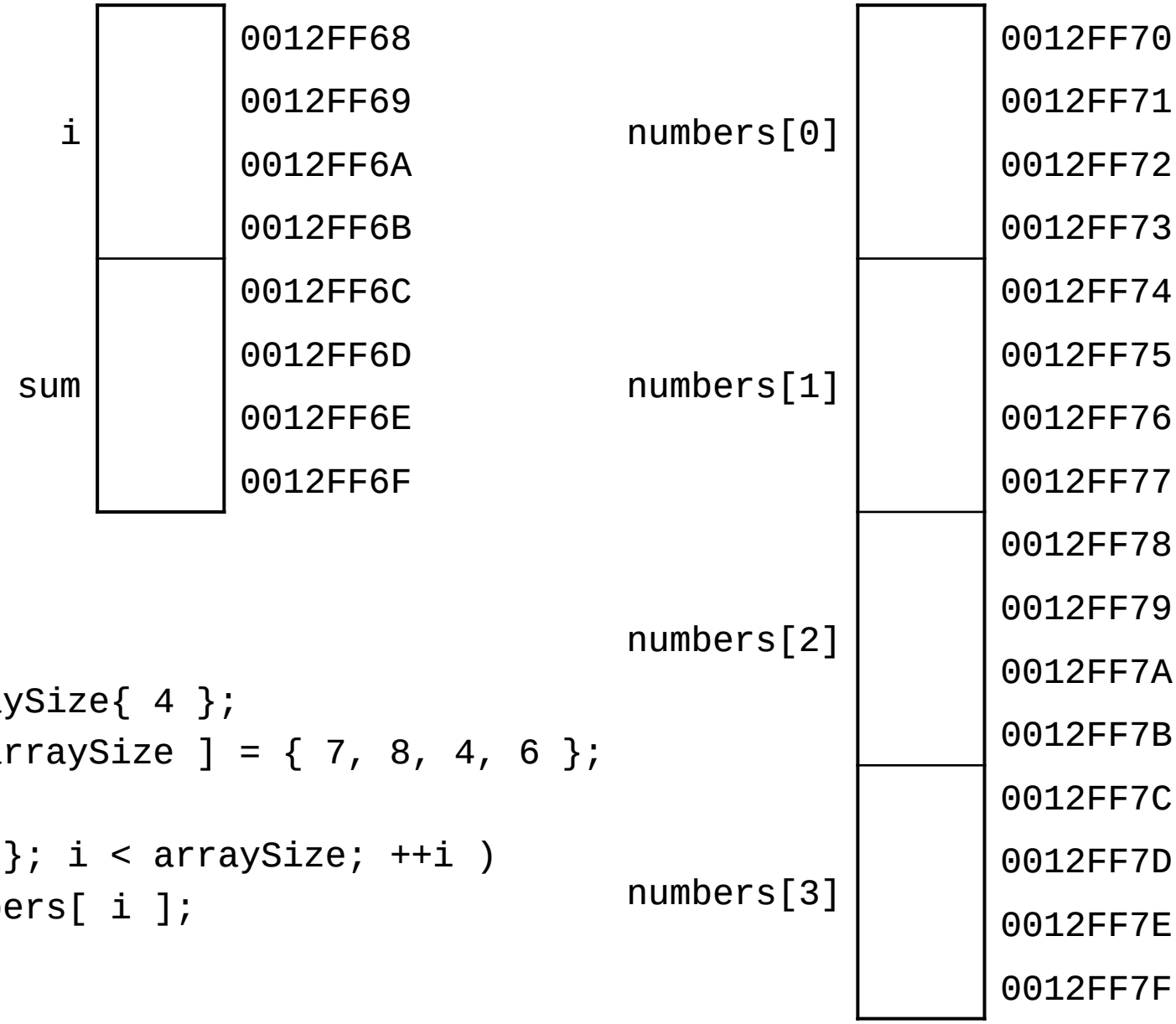



```
int sum{ 0 };
sum += 1;
sum += 2;
sum += 3;
```

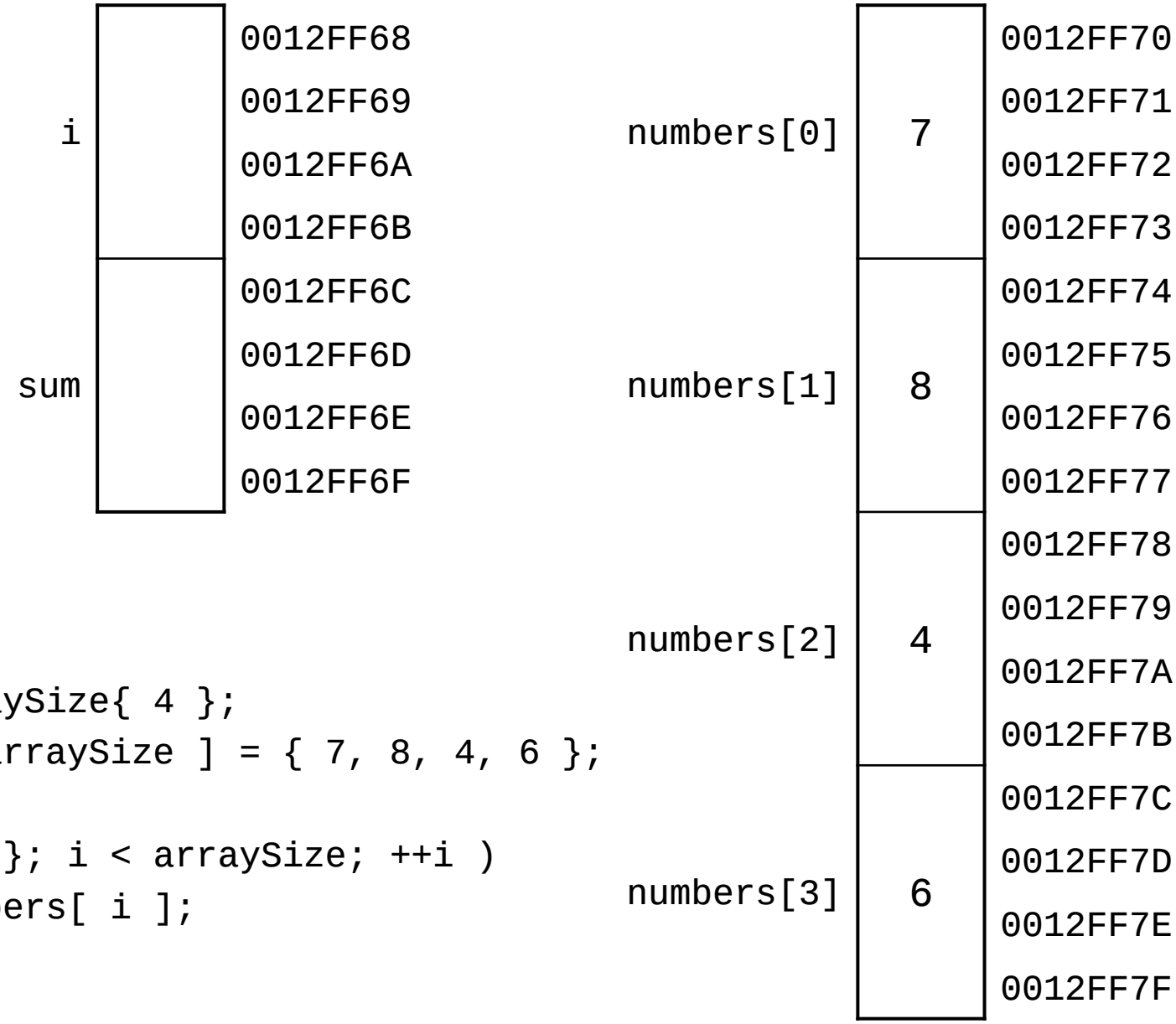
```
#include <iostream>
using namespace std;

int main()
{
   int sum{ 0 };
   for( int i{ 1 }; i <= 3; ++i )
      sum += i;

   cout << "Sum = " << sum << endl;
}
```

```
#include <iostream>
using namespace std;

int main()
{
   const int arraySize{ 4 };
   int numbers[ arraySize ] = { 87, 68, 94, 100 };
   int sum{ 0 };

   for( int i{ 0 }; i < arraySize; ++i )
      sum += numbers[ i ];

   cout << "Sum of array elements: " << sum << endl;
}
```

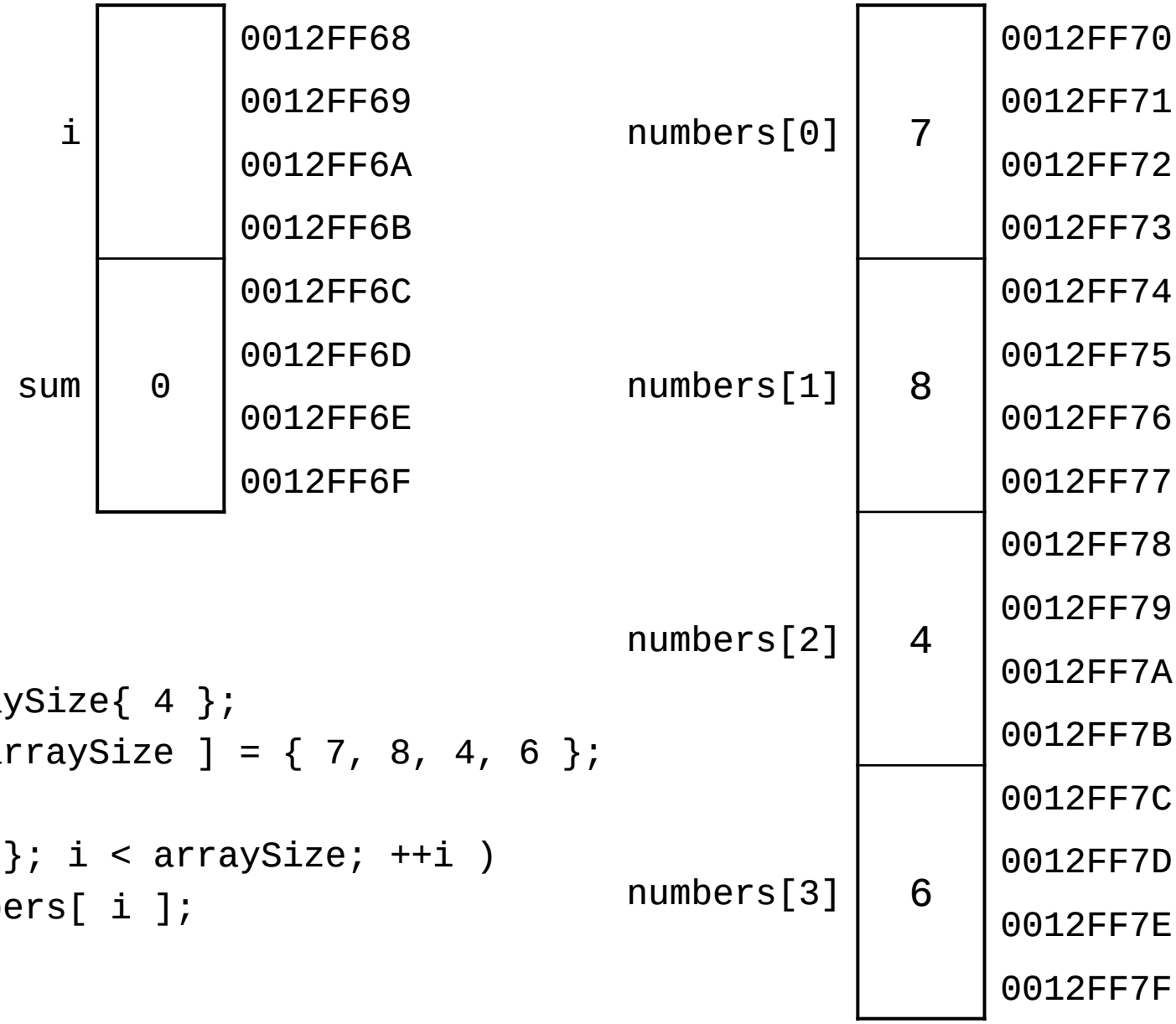
```
Sum of array elements: 349
```



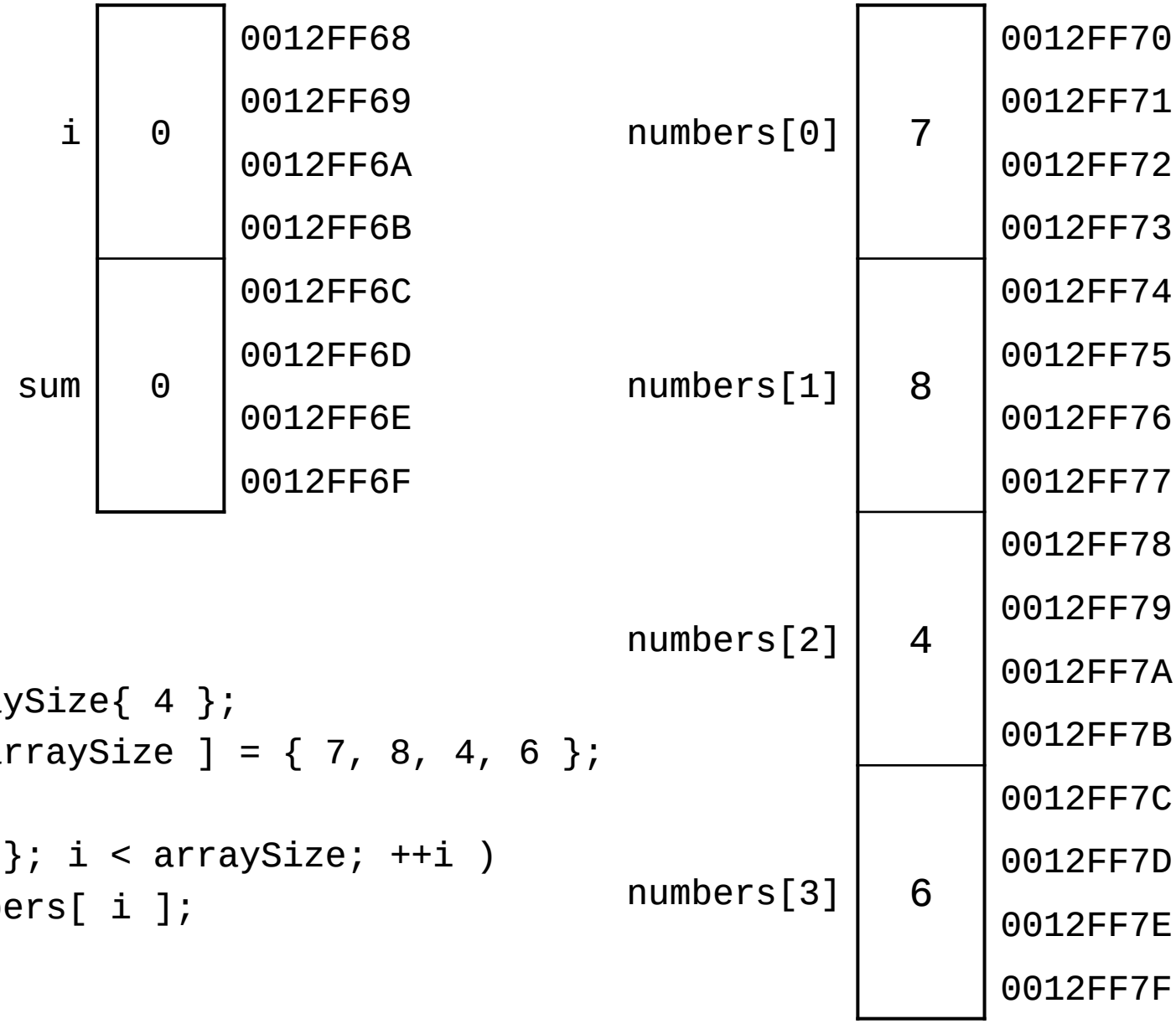

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



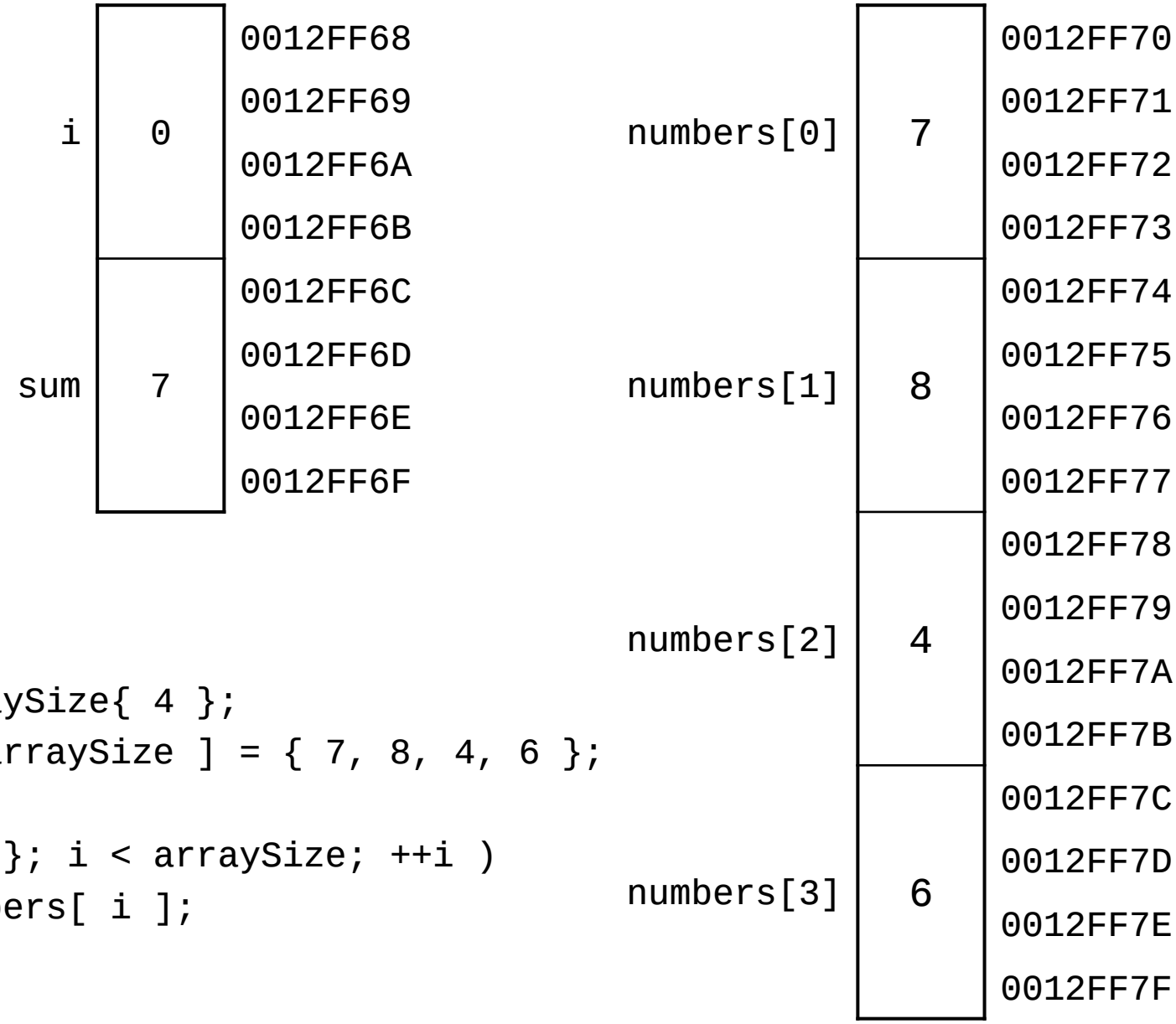

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



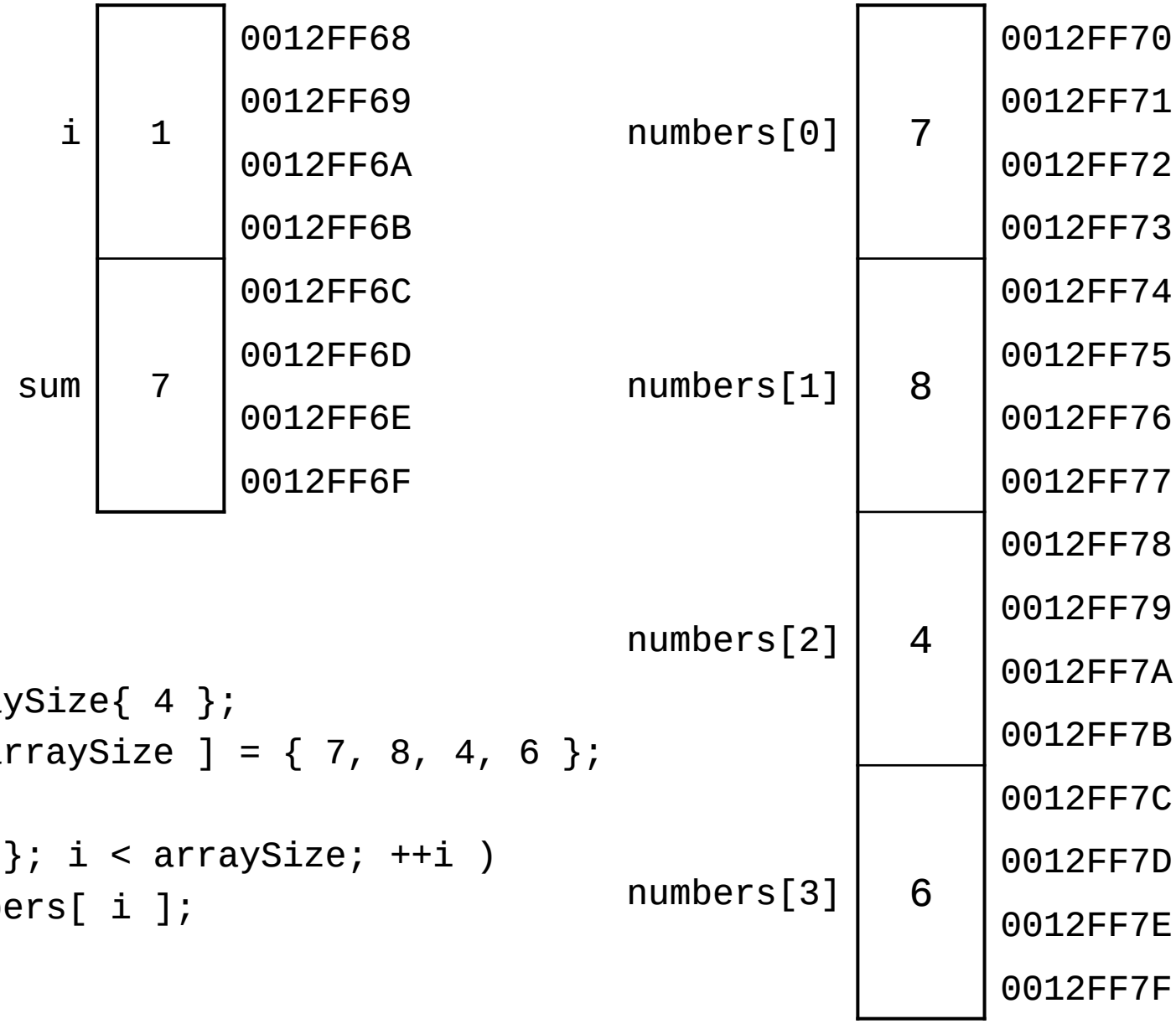

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



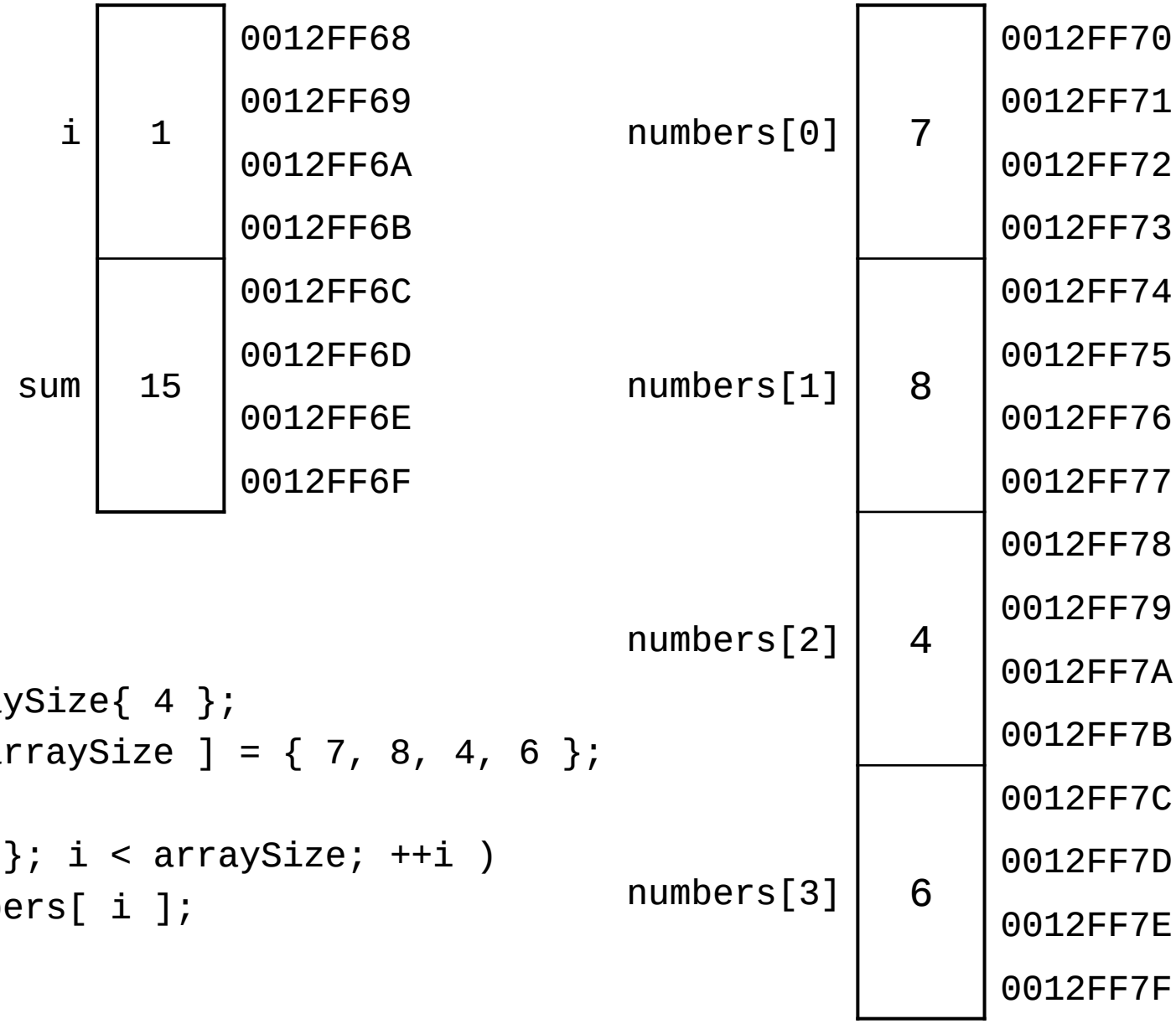

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



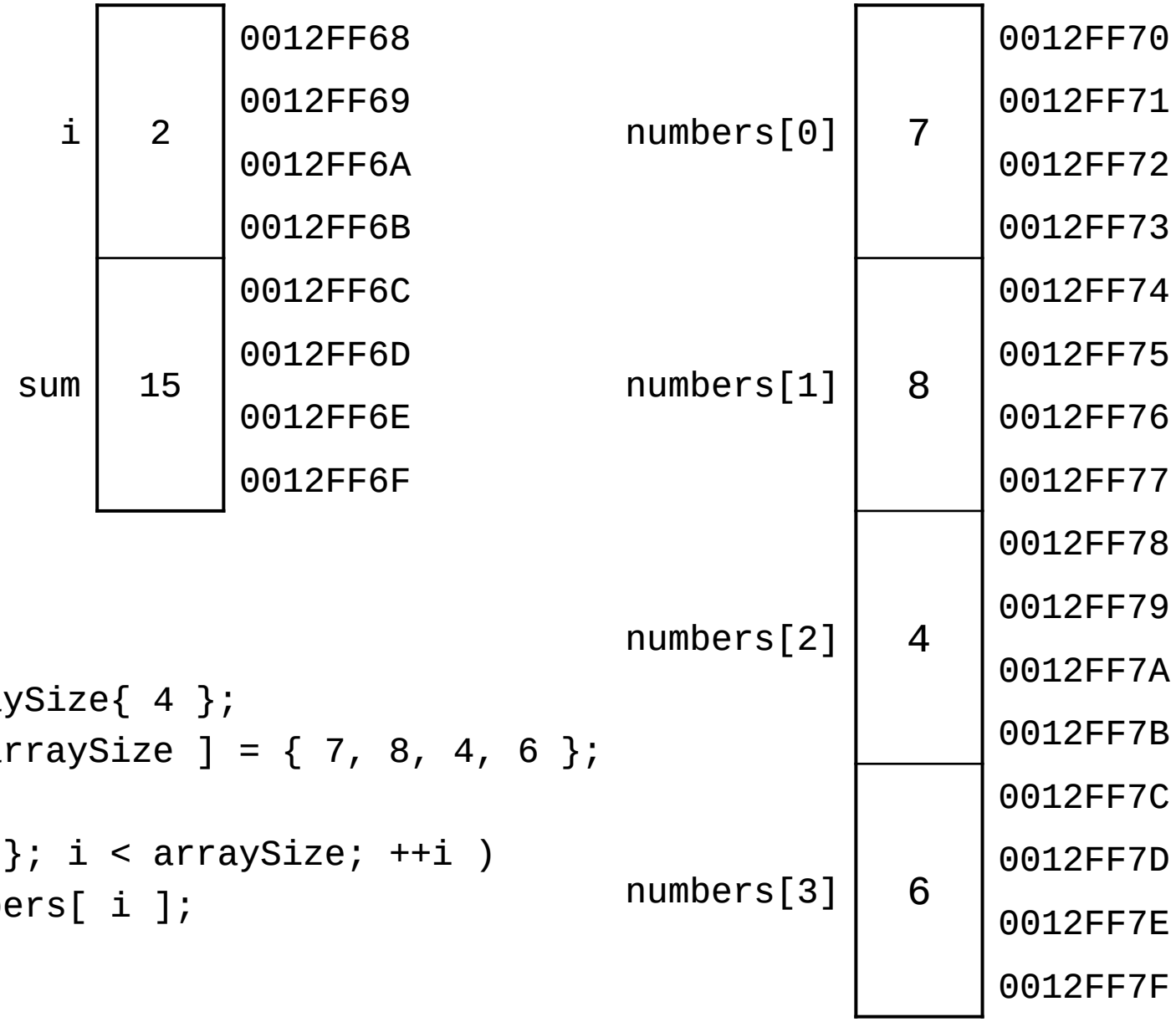

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



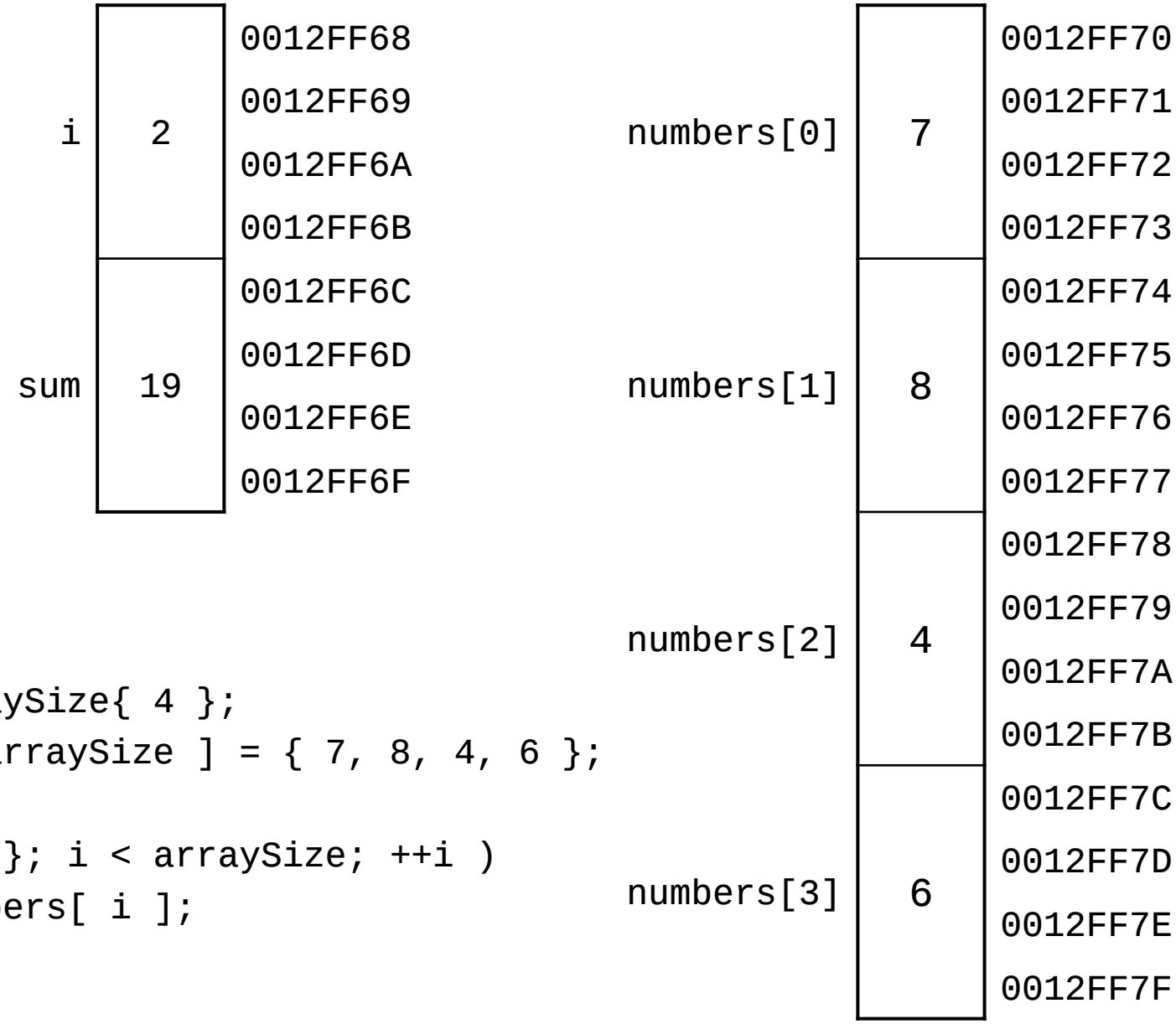

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



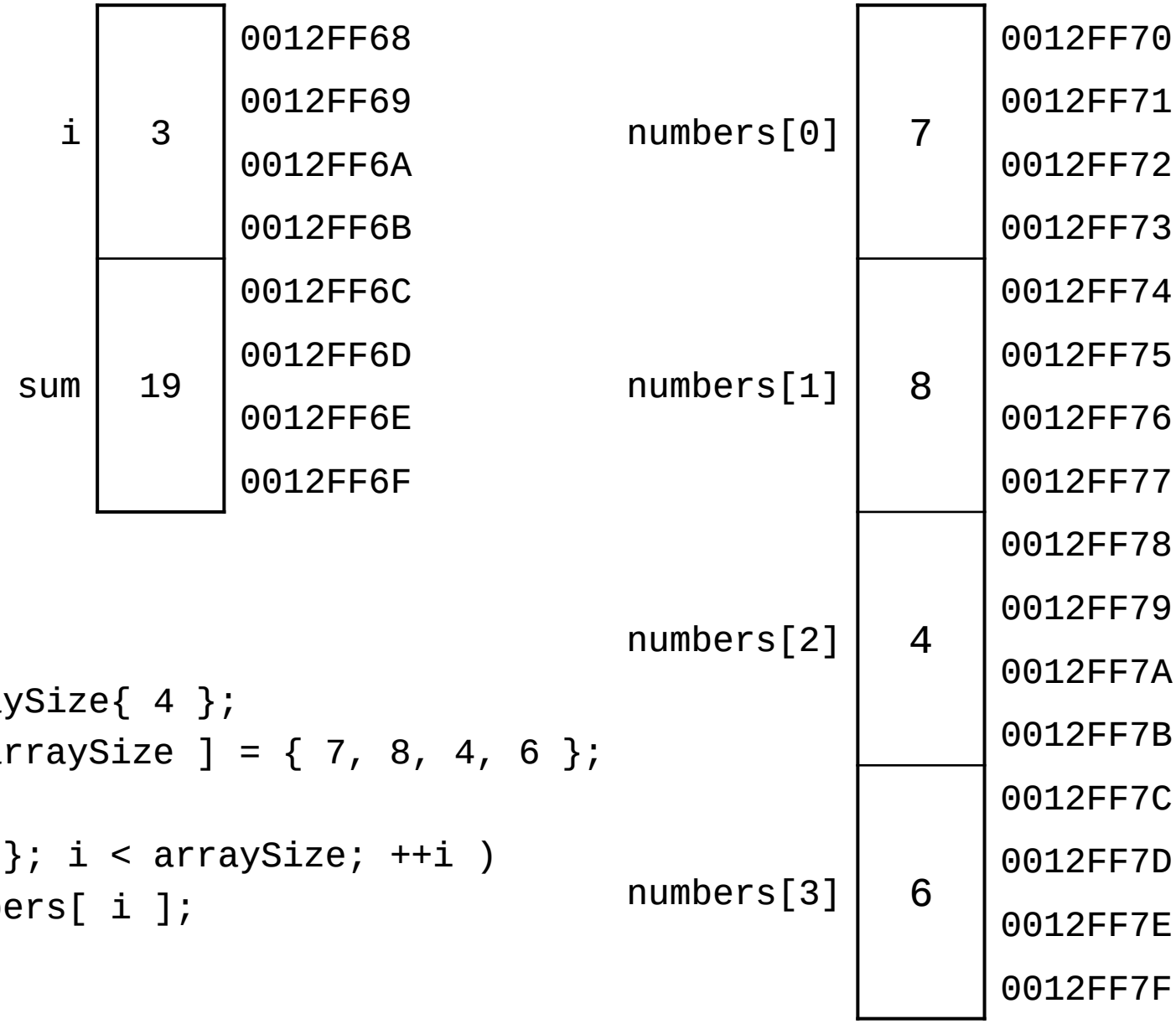

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



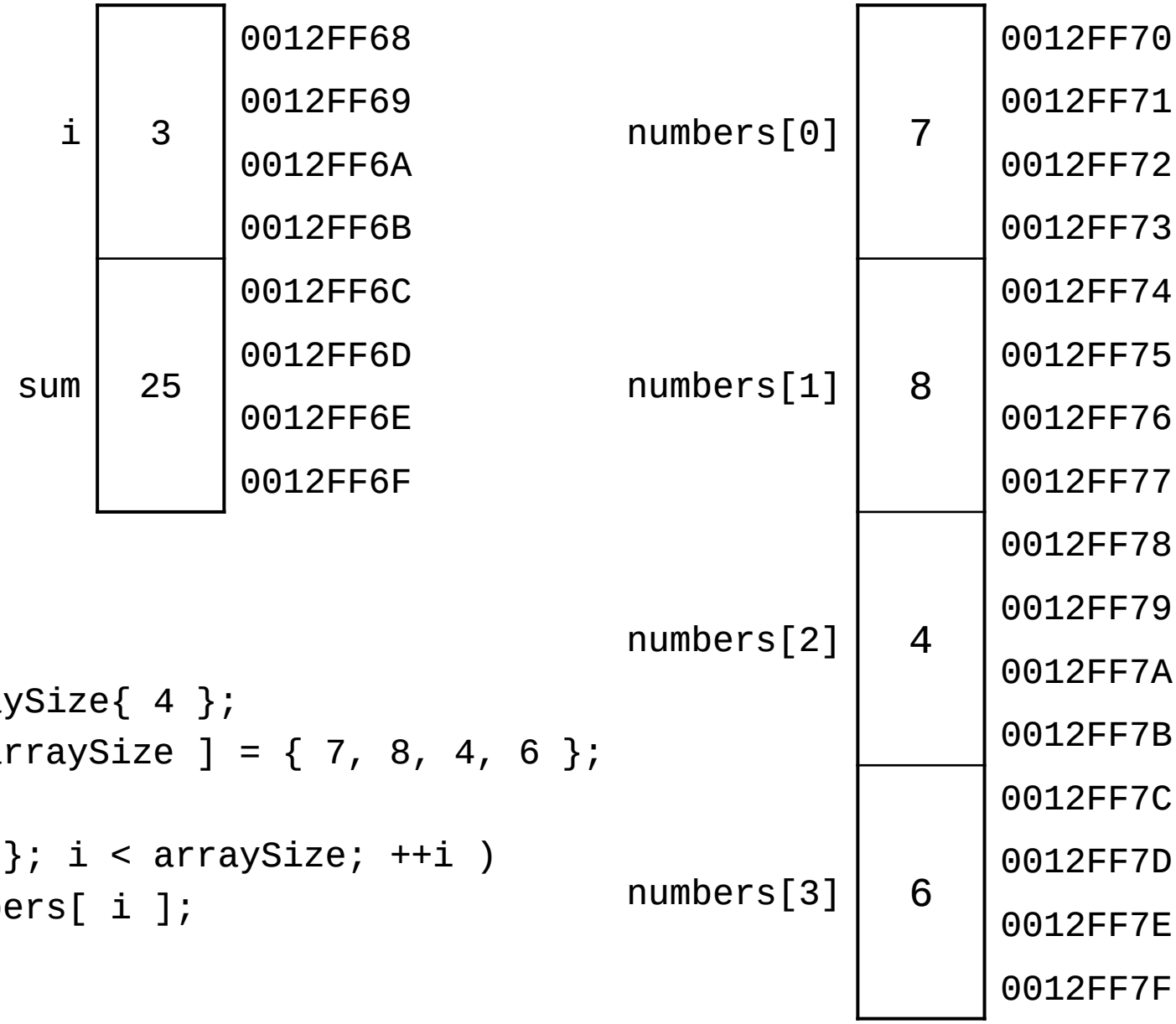

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```



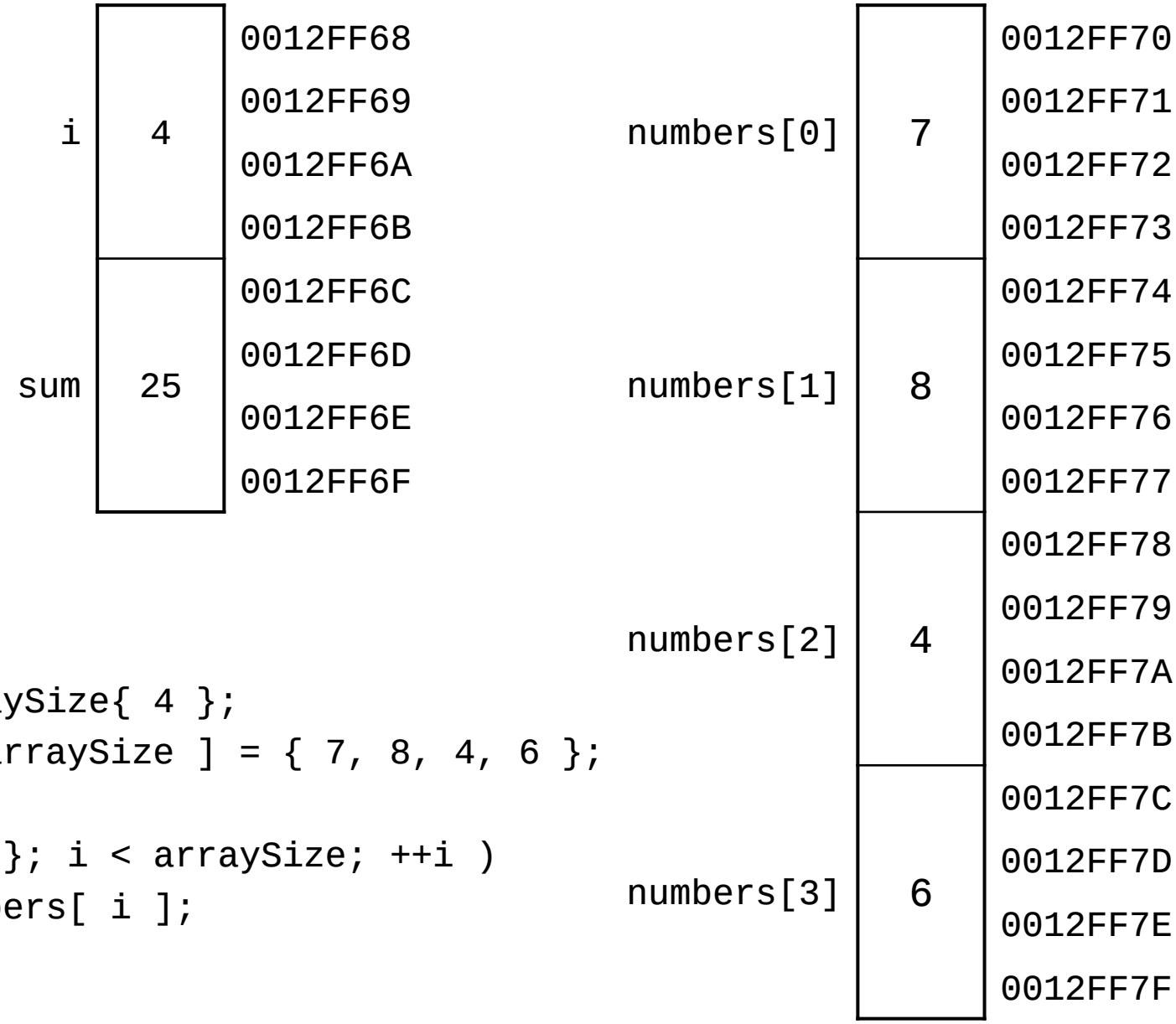

```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```




```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```




```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```




```
const int arraySize{ 4 };
int numbers[ arraySize ] = { 7, 8, 4, 6 };
int sum{ 0 };
for( int i{ 0 }; i < arraySize; ++i )
   sum += numbers[ i ];
```

Range-based for loop

```
#include <iostream>
using namespace std;

int main()
{
   const int arraySize = 4;
   int numbers[ arraySize ] = { 87, 68, 94, 100 };
   int sum{ 0 };

   for( int number: numbers )
      sum += number;

   cout << "Sum of array elements: " << sum << endl;
}
```

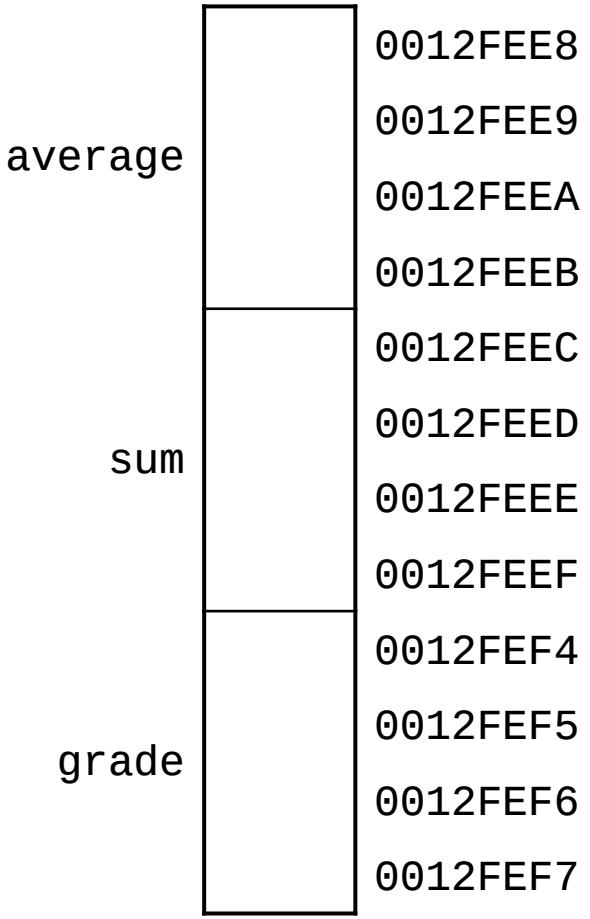
```
Sum of array elements: 349
```

Counter-Controlled Repetition

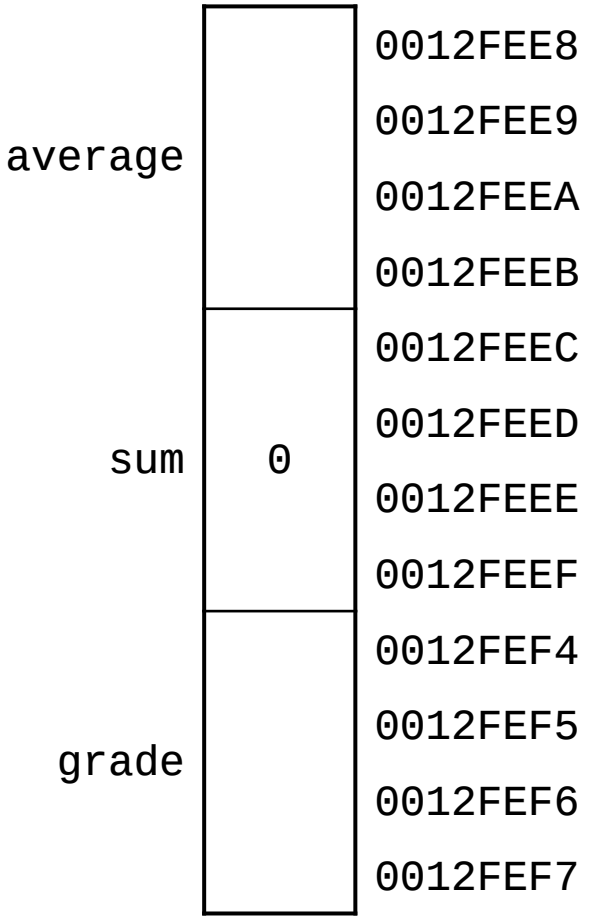
- Definite repetition
  - Number of repetitions known
- Example

  A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Calculate and display the total of all student grades and the class average on the quiz.

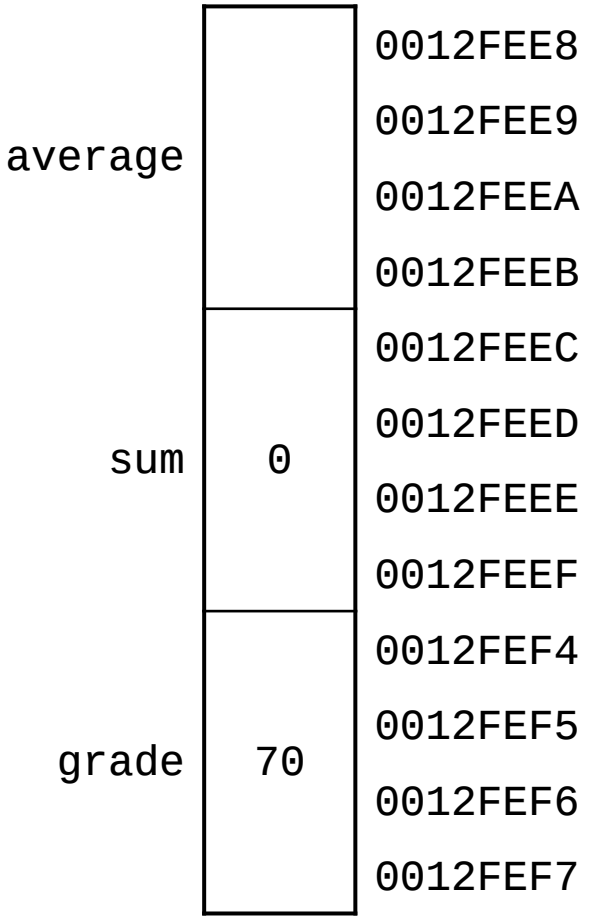
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



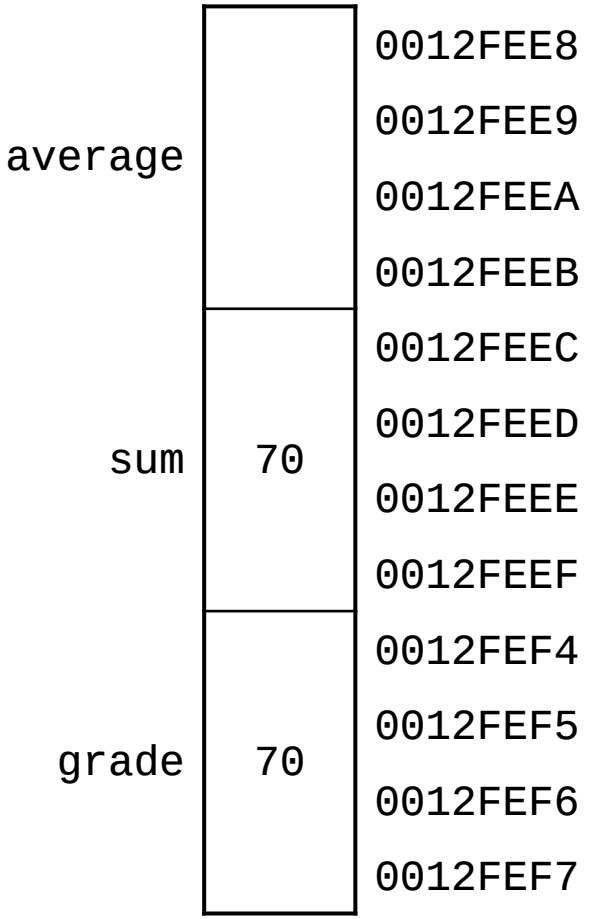
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



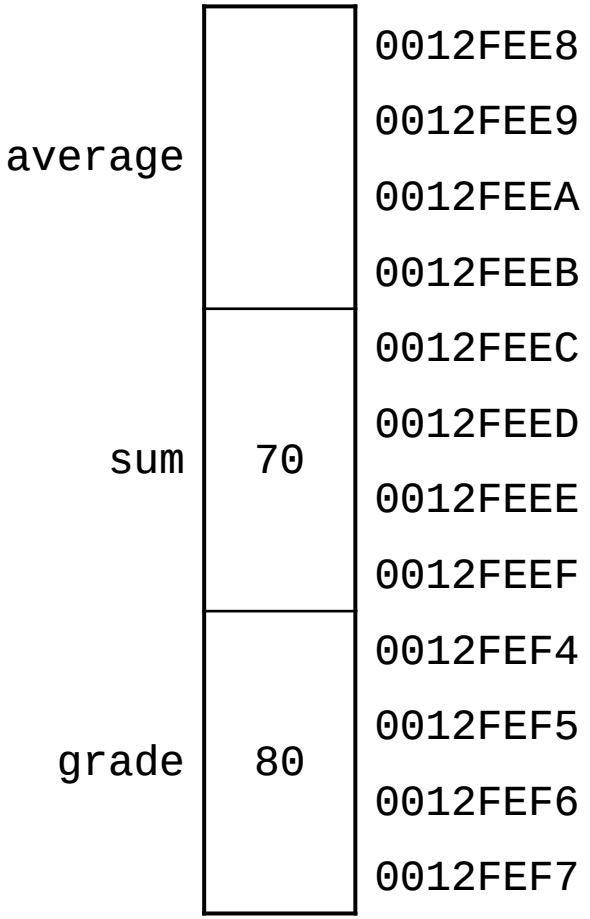
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



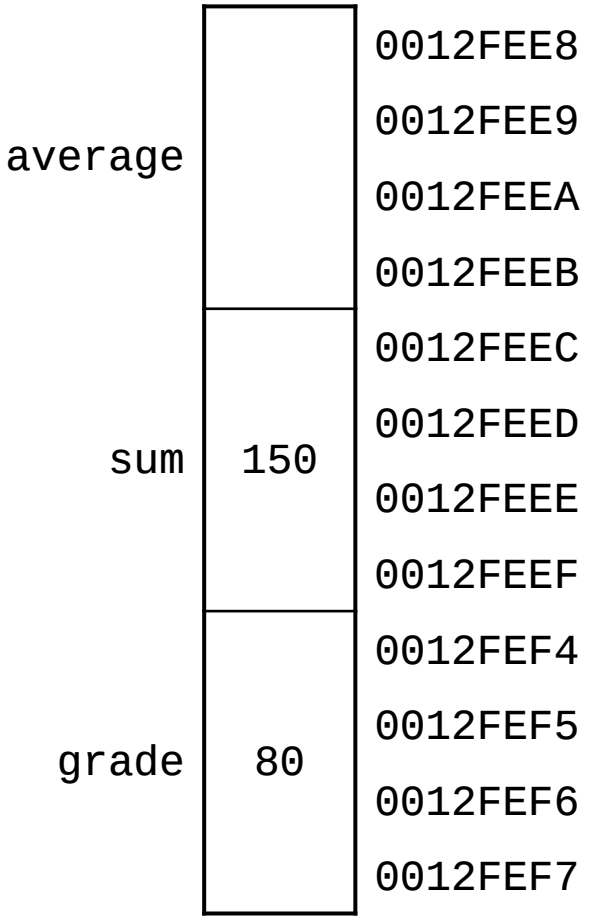
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



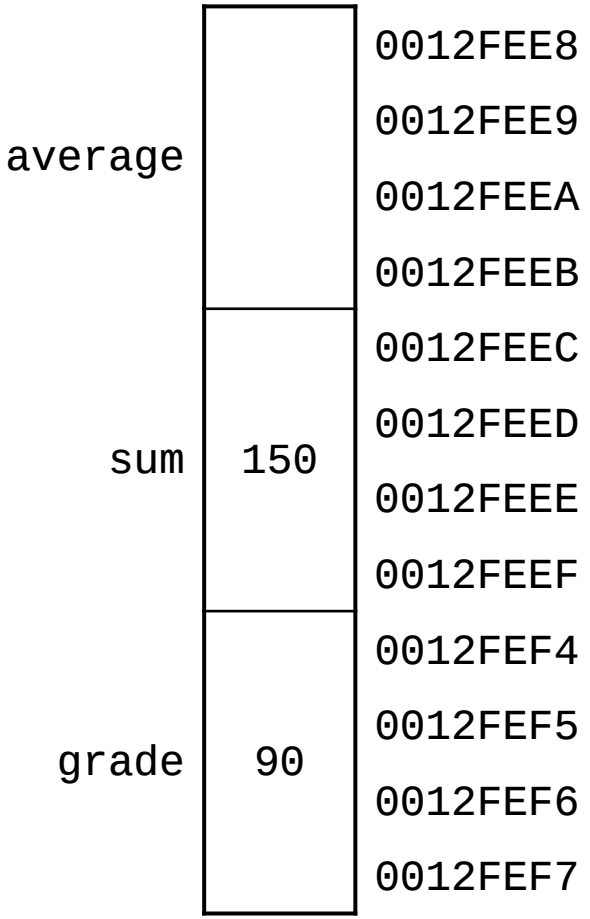
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



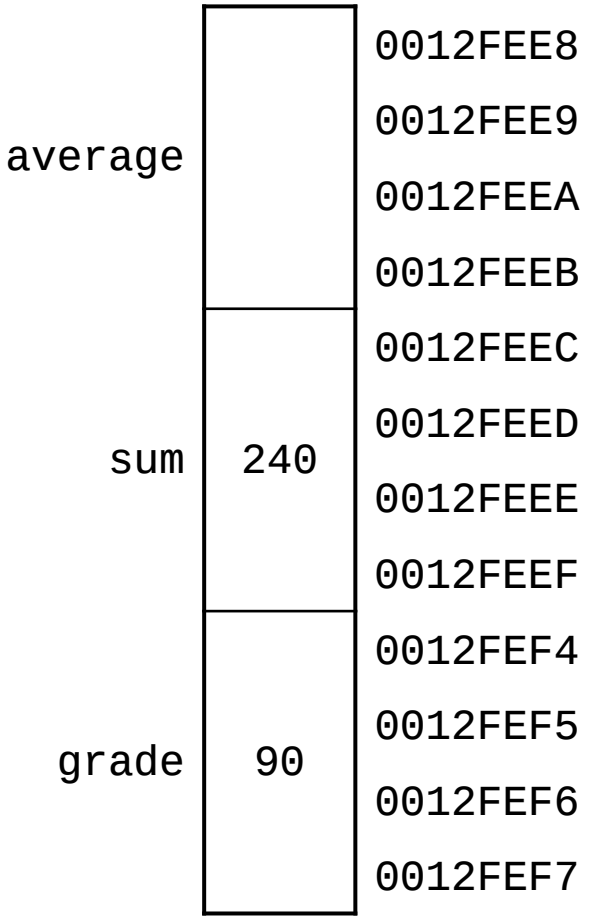
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



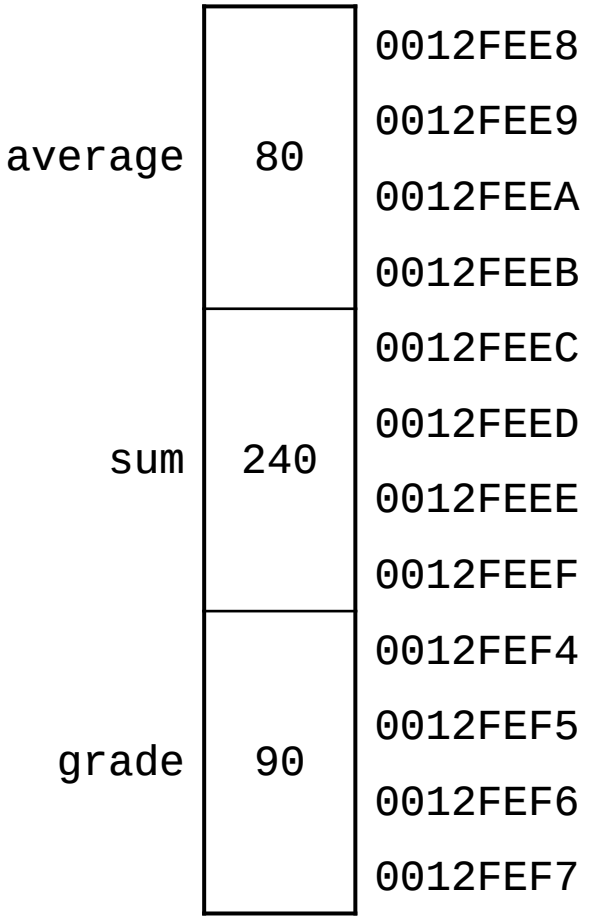
```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```



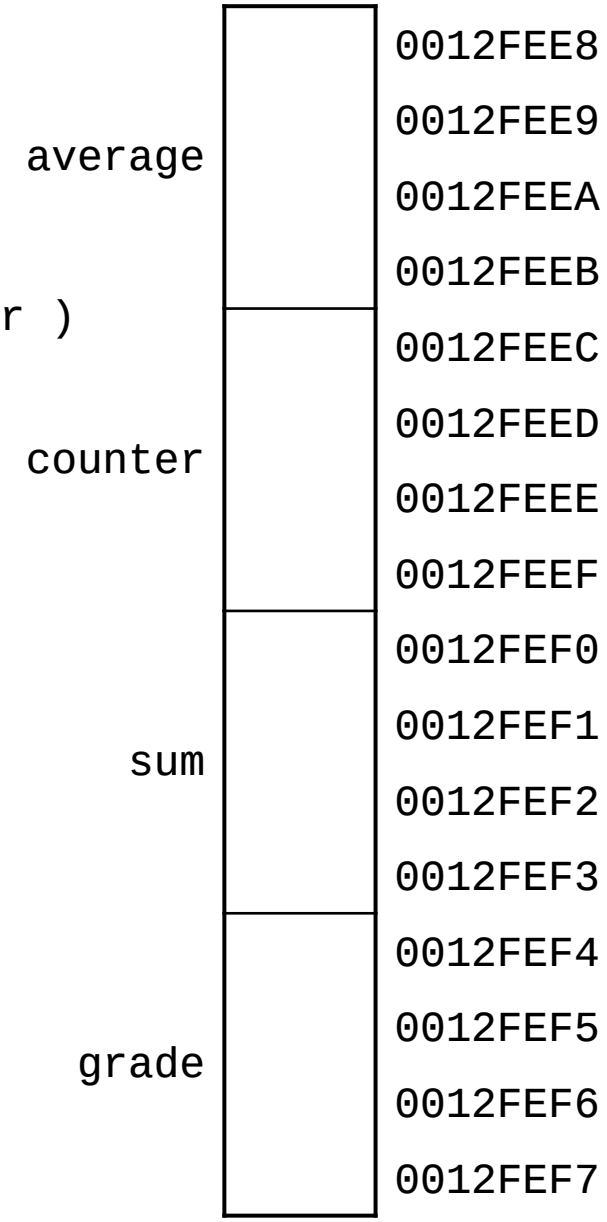
```
int grade;
int sum{ 0 };
for(                    ;                ;            )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```

```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```

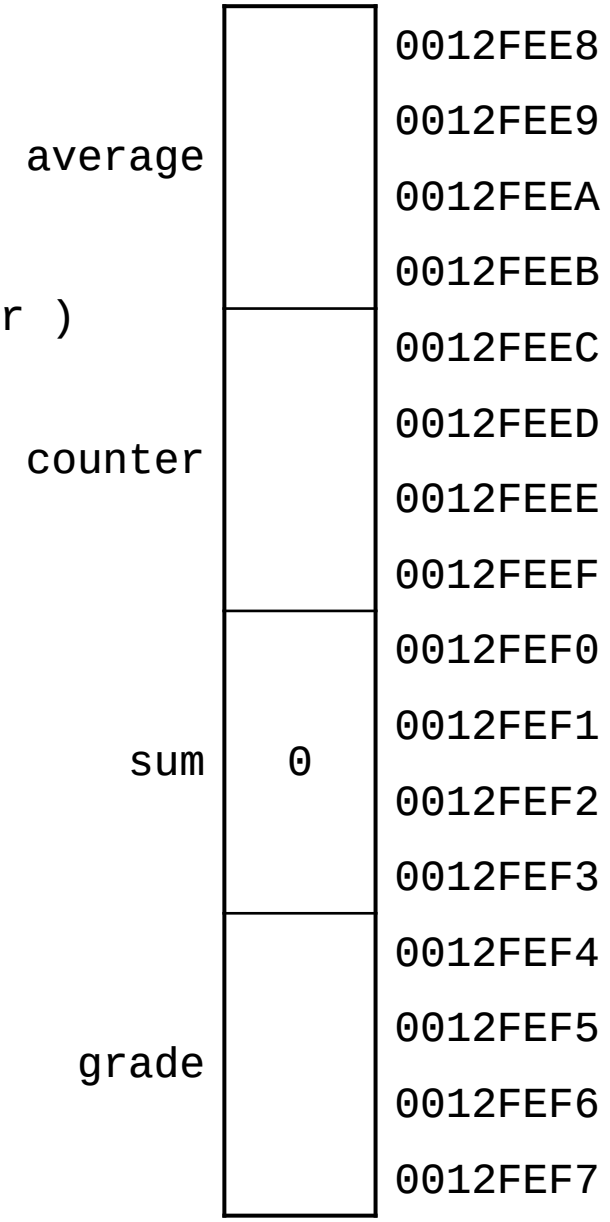
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```

```
int grade;
int sum{ 0 };

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

cout << "Enter grade: ";
cin >> grade;
sum += grade;

int average{ sum / 3 };
```

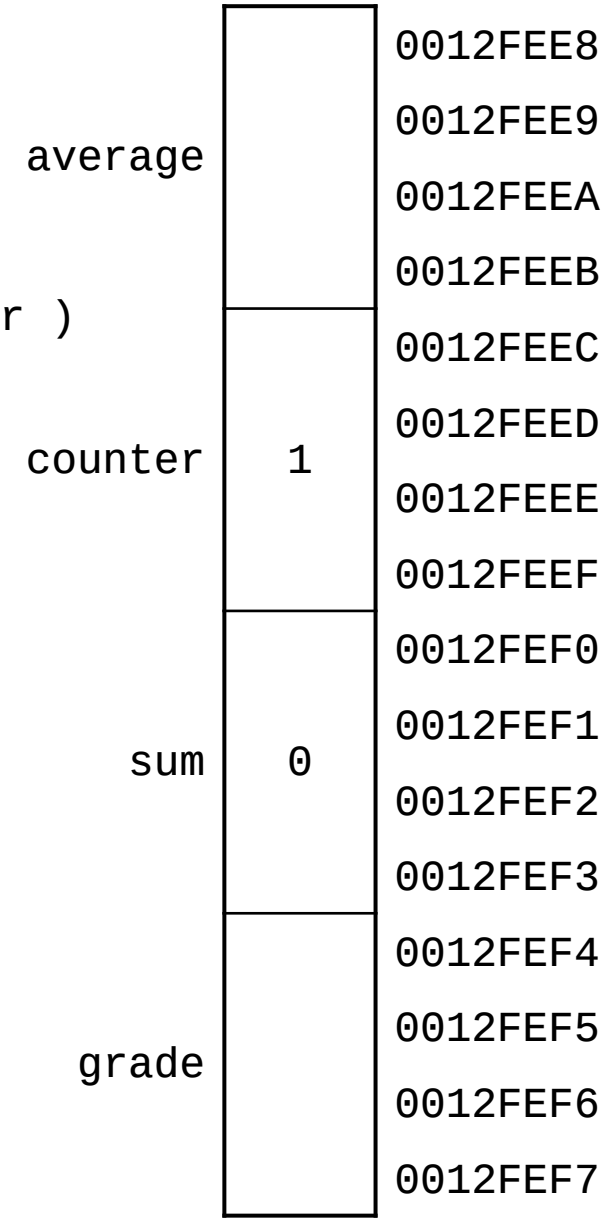
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



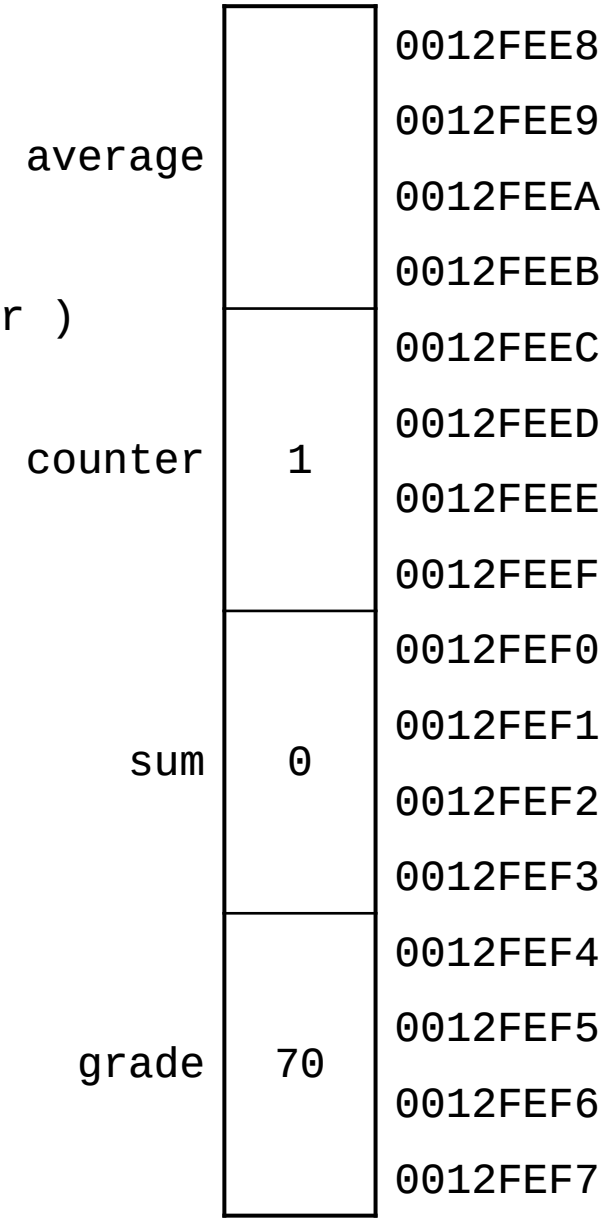
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



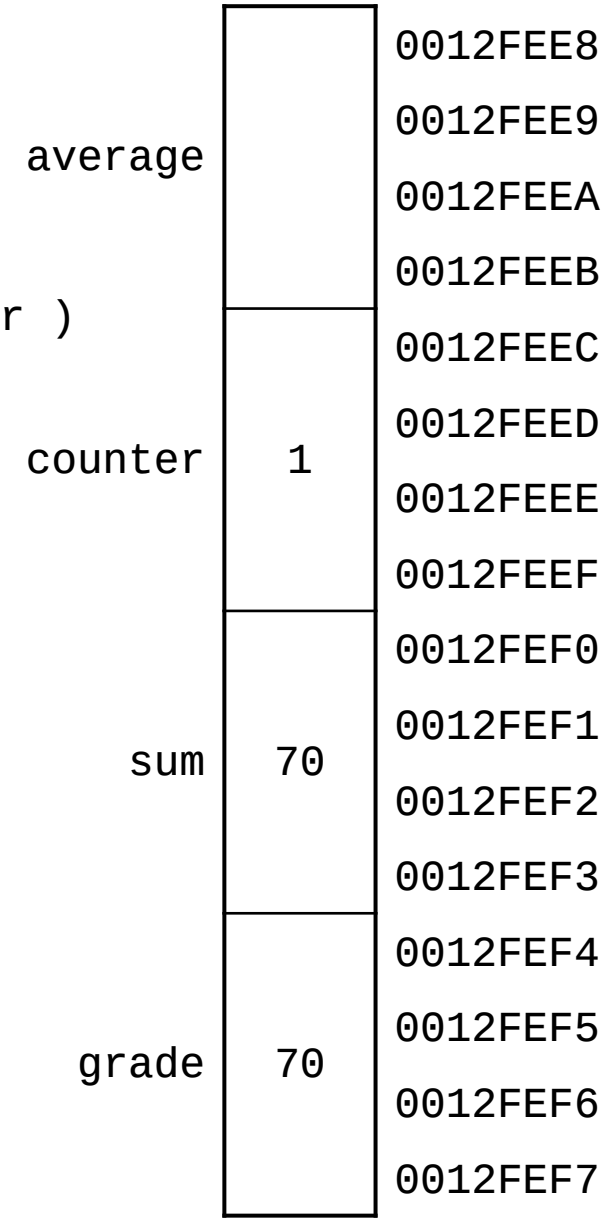
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



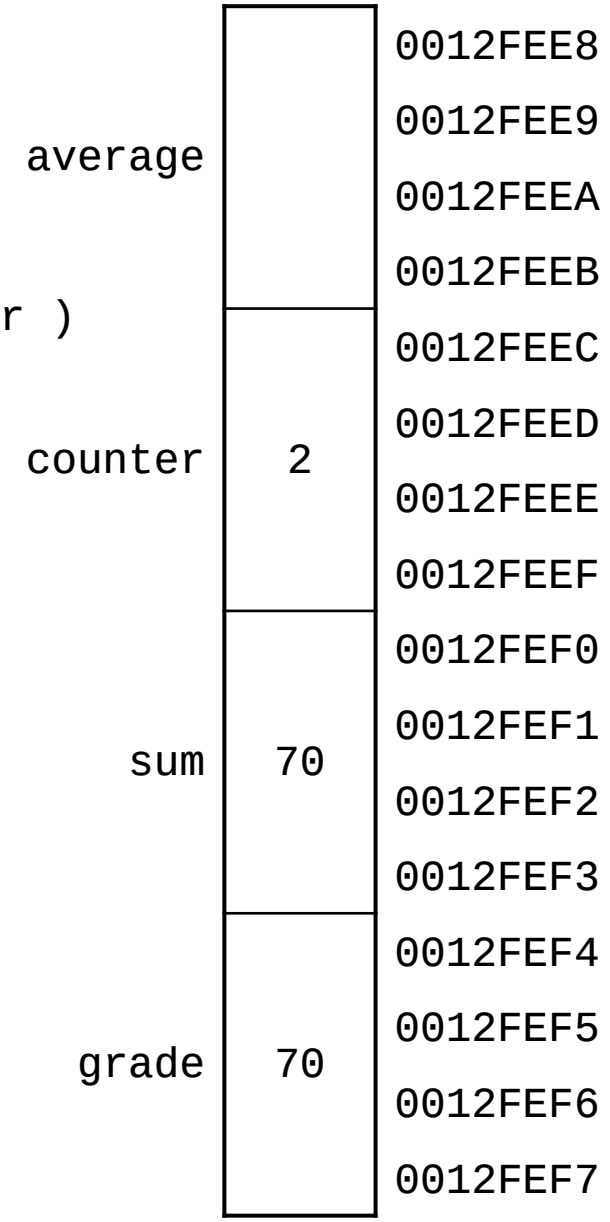
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



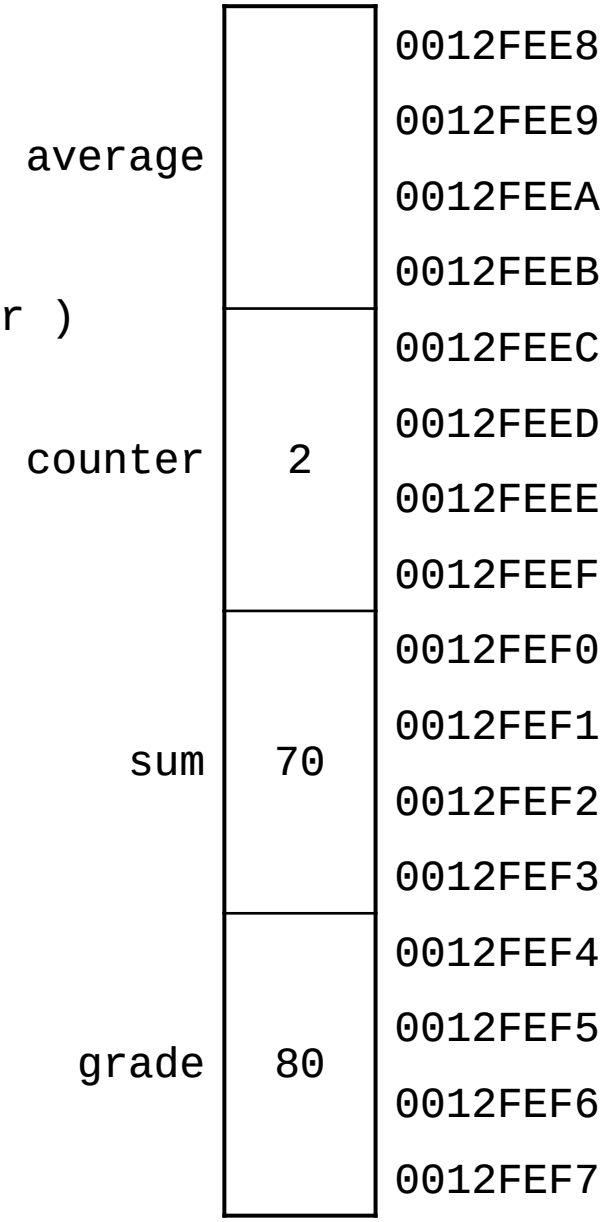
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



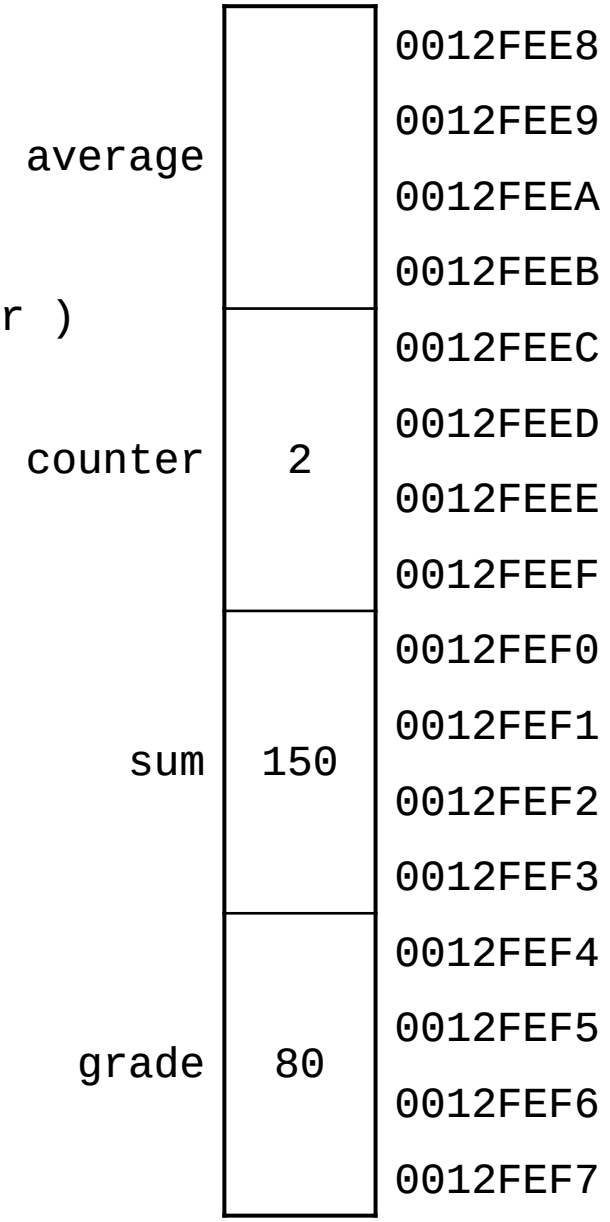
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



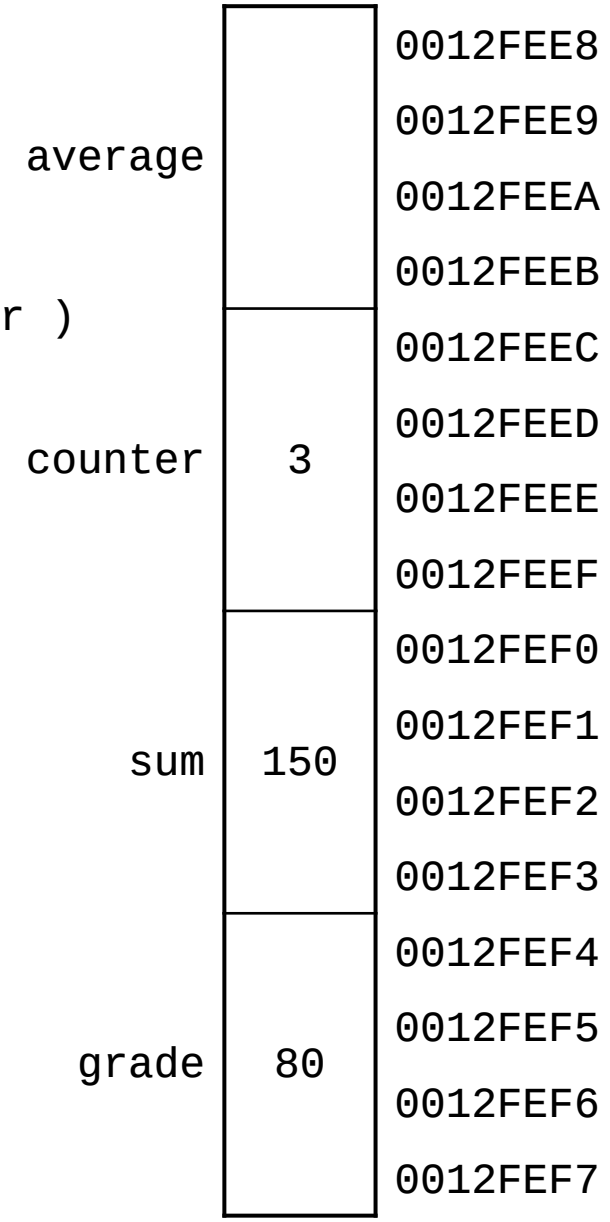
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



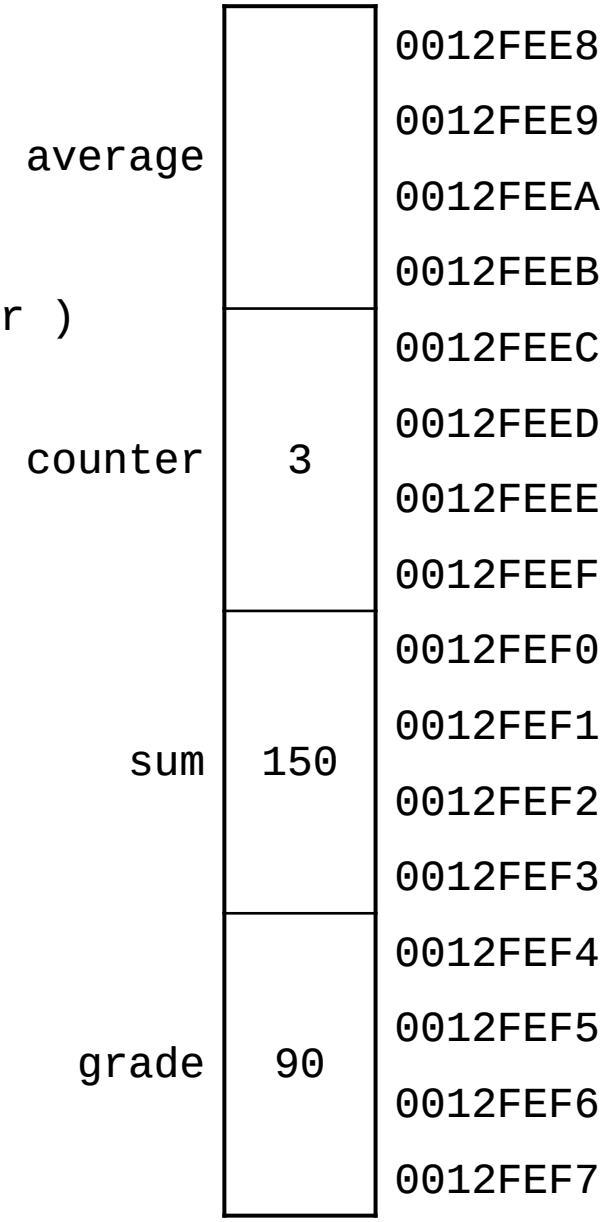
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



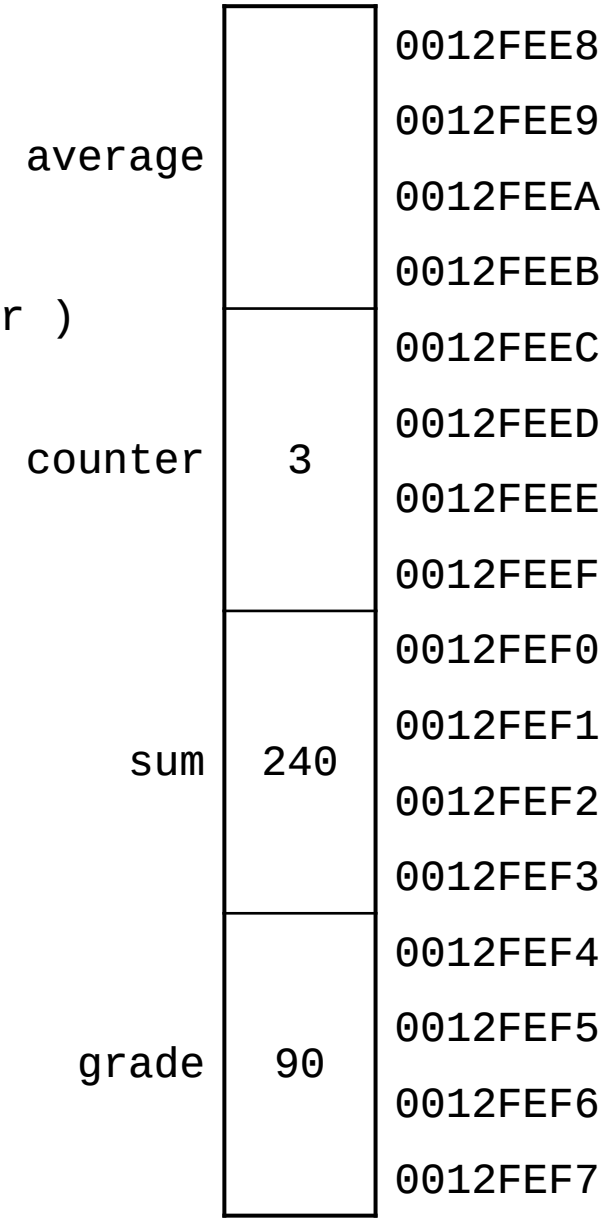
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



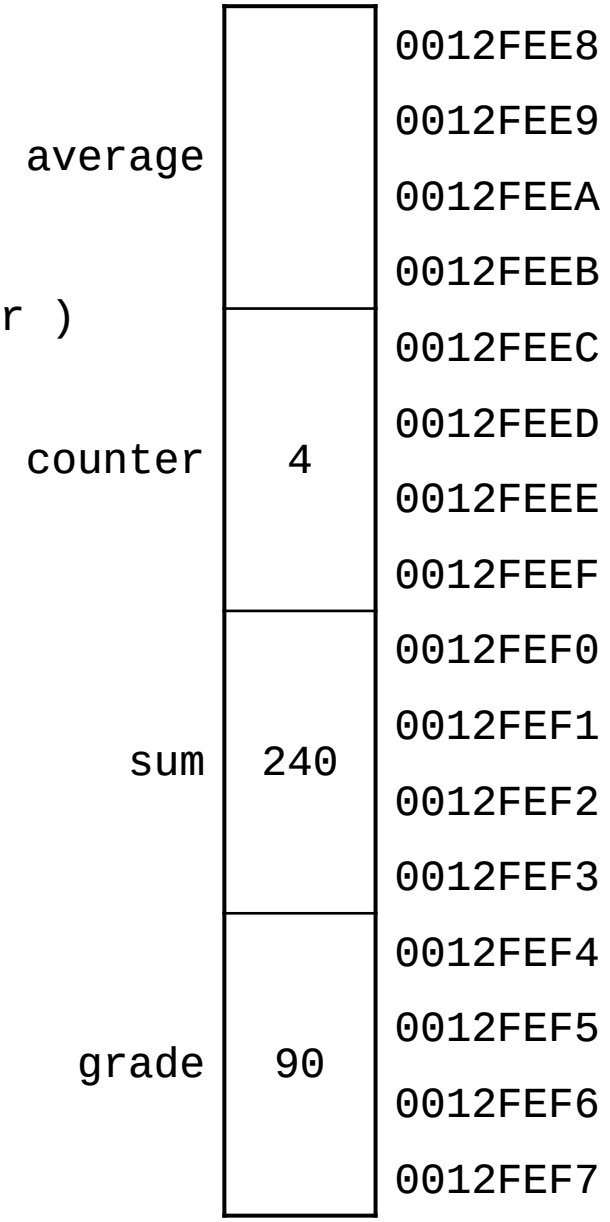
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



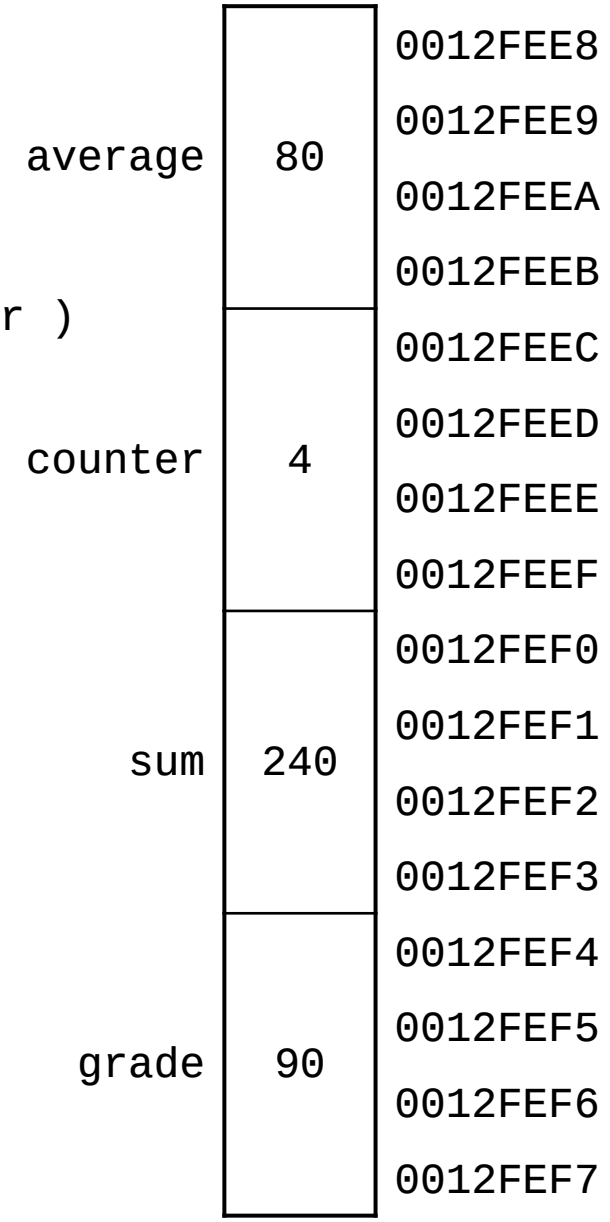
```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



```
int grade;
int sum{ 0 };
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade;
   sum += grade;
}
int average{ sum / 3 };
```



```
#include <iostream>
using namespace std;

int main ()
{
   int grade[ 11 ];
   int sum{ 0 };
   for( int counter{ 1 }; counter <= 10; ++counter )
   {
      cout << "Enter grade: ";
      cin >> grade[ counter ];
      sum += grade[ counter ];
   }

   int average{ sum / 3 }; // average of grades

   cout << "\nThe sum of all 3 grades is " << sum << endl;
   cout << "Class average is " << average << endl;
}
```

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]		0012FF68
grade[2]		0012FF6C
grade[3]		0012FF70
counter		0012FF74
sum		0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]		0012FF68
grade[2]		0012FF6C
grade[3]		0012FF70
counter	1	0012FF74
sum	0	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]		0012FF6C
grade[3]		0012FF70
counter	1	0012FF74
sum	0	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]		0012FF6C
grade[3]		0012FF70
counter	1	0012FF74
sum	70	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]		0012FF6C
grade[3]		0012FF70
counter	2	0012FF74
sum	70	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]		0012FF70
counter	2	0012FF74
sum	70	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]		0012FF70
counter	2	0012FF74
sum	150	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]		0012FF70
counter	3	0012FF74
sum	150	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]	90	0012FF70
counter	3	0012FF74
sum	150	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]	90	0012FF70
counter	3	0012FF74
sum	240	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average		0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]	90	0012FF70
counter	4	0012FF74
sum	240	0012FF78

```
int sum{ 0 };
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
   sum += grade[ counter ];
}
int average{ sum / 3 };
```

average	80	0012FF60
grade[0]		0012FF64
grade[1]	70	0012FF68
grade[2]	80	0012FF6C
grade[3]	90	0012FF70
counter	4	0012FF74
sum	240	0012FF78

```
#include <iostream>
using namespace std;

int main ()
{
   int grade[ 11 ];
   for( int counter{ 1 }; counter <= 10; ++counter )
   {
      cout << "Enter grade: ";
      cin >> grade[ counter ];
   }

   int sum{ 0 };
   for( int i{ 1 }; i <= 3; i++ )
      sum += grade[ i ];

   int average{ sum / 3 }; // average of grades

   cout << "\nThe sum of all 3 grades is " << sum << endl;
   cout << "Class average is " << average << endl;
}
```

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter		0012FF6C
grade[0]		0012FF70
grade[1]		0012FF74
grade[2]		0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	1	0012FF6C
grade[0]		0012FF70
grade[1]		0012FF74
grade[2]		0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	1	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]		0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	2	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]		0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	2	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]		0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum		0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum	0	0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum	70	0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum	150	0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average		0012FF64
sum	240	0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

```
int grade[ 4 ];
for( int counter{ 1 }; counter <= 3; ++counter )
{
   cout << "Enter grade: ";
   cin >> grade[ counter ];
}

int sum{ 0 };
for( int i{ 1 }; i <= 3; i++ )
   sum += grade[ i ];
int average{ sum / 3 };
```

average	80	0012FF64
sum	240	0012FF68
counter	3	0012FF6C
grade[0]		0012FF70
grade[1]	70	0012FF74
grade[2]	80	0012FF78
grade[3]	90	0012FF7C

Indefinite Repetition

```
#include <iostream>
using namespace std;

int main ()
{
   int grade;
   int sum{ 0 };
   int counter{ 0 };

   cout << "Enter grade: ";
   while( cin >> grade )
   {
      sum += grade;
      ++counter;
      cout << "Enter grade: ";
   }

   int average{ sum / counter }; // average of grades

   cout << "\nThe sum of all " << counter << " grades is "
        << sum << endl;
   cout << "Class average is " << average << endl;
}
```

Indefinite Repetition

Using the End-Of-File (EOF) which is Ctrl + Z for windows to stop the reading from standard input.

```
int n;
while( cin >> n )
   cout << n << endl;
```

`cin >> n` returns false when encounters End-Of-File.

```
Enter grade: 75
Enter grade: 94
Enter grade: 97
Enter grade: 88
Enter grade: 70
Enter grade: 64
Enter grade: 83
Enter grade: 89
Enter grade: ^Z

The sum of all 8 grades is 660
Class average is 82
```

Sentinel-Controlled Repetition

- Let's generalize the class average problem

  Develop a class-averaging program that will process an arbitrary number of grades each time the program is run.
- The program must process an arbitrary number of grades.
  - How can the program determine when to stop the input of grades?
- Can use a special value called a sentinel value (also called a signal value, a dummy value or a flag value) to indicate "end of data entry."
- The sentinel value must not be an acceptable input value

```
Enter grade or -1 to quit: 75
Enter grade or -1 to quit: 94
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 70
Enter grade or -1 to quit: 64
Enter grade or -1 to quit: 83
Enter grade or -1 to quit: 89
Enter grade or -1 to quit: -1

Total of all 8 grades entered is 660
Class average is 82.50
```

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int grade;
   int sum{ 0 };
   int counter{ 0 };
   while( grade != -1 )
   {
      cout << "Enter grade or -1 to quit: ";
      cin >> grade;
      sum += grade;
      ++counter;
   }
   int average{ sum / counter }; // average of grades

   cout << "\nThe sum of all " << counter << " grades is "
        << sum << endl;
   cout << "Class average is " << average << endl;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int grade;
   int sum{ 0 };
   int counter{ 0 };
   cout << "Enter grade or -1 to quit: ";
   cin >> grade;
   while( grade != -1 )
   {
      sum += grade;
      ++counter;
      cout << "Enter grade or -1 to quit: ";
      cin >> grade;
   }
   int average{ sum / counter }; // average of grades

   cout << "\nThe sum of all " << counter << " grades is "
        << sum << endl;
   cout << "Class average is " << average << endl;
}
```

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int grade;
   int sum{ 0 };
   int counter{ 0 };

   cout << "Enter grade or -1 to quit: ";
   cin >> grade;
   while( grade != -1 )
   {
      sum += grade;
      ++counter;

      cout << "Enter grade or -1 to quit: ";
      cin >> grade;
   }
```

```
   double average;
   if( counter != 0 )
   {
      average = static_cast< double >( sum ) / counter;

      cout << "\nThe sum of all " << counter << " grades entered is "
           << sum << endl;
      cout << "Class average is "
           << setprecision( 2 ) << fixed << average << endl;
   }
   else
      cout << "No grades were entered\n";
}
```

```
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 72
Enter grade or -1 to quit: -1

Total of all 3 grades entered is 257
Class average is 85.67
```

Formatting the Floating-Point Numbers

```
average = 1234.5678;
cout << setprecision( 2 ) << fixed << average << endl;
cout << setprecision( 4 ) << fixed << average << endl;
cout << setprecision( 6 ) << fixed << average << endl;
```

```
1234.56
1234.5678
1234.567800
```

Formatting the Floating-Point Numbers

```
average = 1234.5678;
cout << setprecision( 2 ) << average << endl;
cout << setprecision( 4 ) << average << endl;
cout << setprecision( 6 ) << average << endl;
cout << setprecision( 8 ) << average << endl;
cout << setprecision( 10 ) << average << endl;
```

```
1.2e+003
1234
1234.56
1234.5678
1234.5678
```

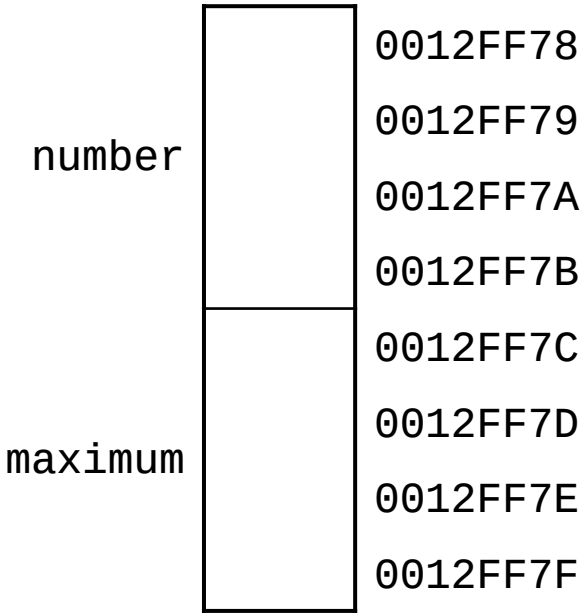
Data types
long double (8 bytes) 絕對值範圍大約是 $2.2 \cdot 10^{-308} \sim 1.8 \cdot 10^{308}$
double (8 bytes) 絕對值範圍大約是 $2.2 \cdot 10^{-308} \sim 1.8 \cdot 10^{308}$
float (4 bytes) 絕對值範圍大約是 $1.2 \cdot 10^{-38} \sim 3.4 \cdot 10^{38}$
unsigned long long int (unsigned long long) (8 bytes) $0 \sim 2^{64}-1$
long long int (long long) (8 bytes) $-2^{63} \sim 2^{63}-1$
unsigned long int (unsigned long) (4 bytes) $0 \sim 2^{32}-1$
long int (long) (4 bytes) $-2^{31} \sim 2^{31}-1$ (2147483647)
unsigned int (unsigned) (4 bytes) $0 \sim 2^{32}-1$ (4294967295)
int (4 bytes) $-2^{31} \sim 2^{31}-1$ (2147483647)
unsigned short int (unsigned short) (2) $0 \sim 2^{16}-1$ (65535)
short int (short) (2 bytes) $-2^{15} \sim 2^{15}-1$ (32767)
unsigned char (1 byte) $0 \sim 2^{8}-1$ (0 ~ 255)
char (1 byte) $-2^{7} \sim 2^{7}-1$ (-128 ~ 127)
bool (1 byte) (false becomes 0, true becomes 1)

Fig 5.5 Promotion hierarchy for fundamental data types

Maximum Finding

```
cin >> maximum;
cin >> number;
if( number > maximum )
    maximum = number;

cin >> number;
if( number > maximum )
    maximum = number;

cin >> number;
if( number > maximum )
    maximum = number;
```

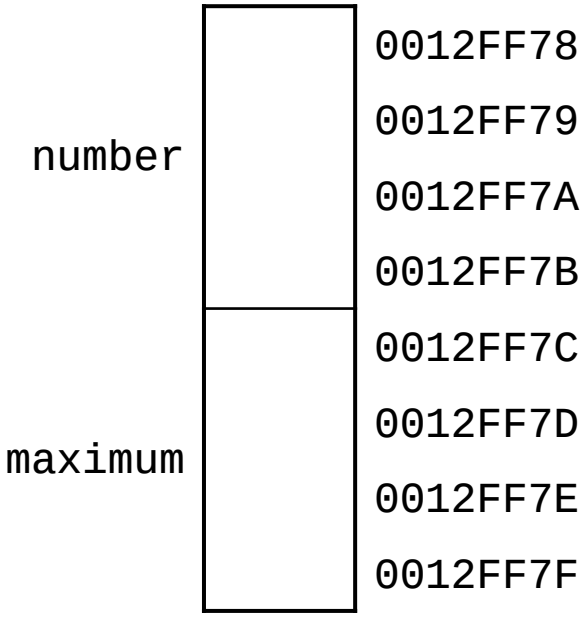
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for(      ;       ;     )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

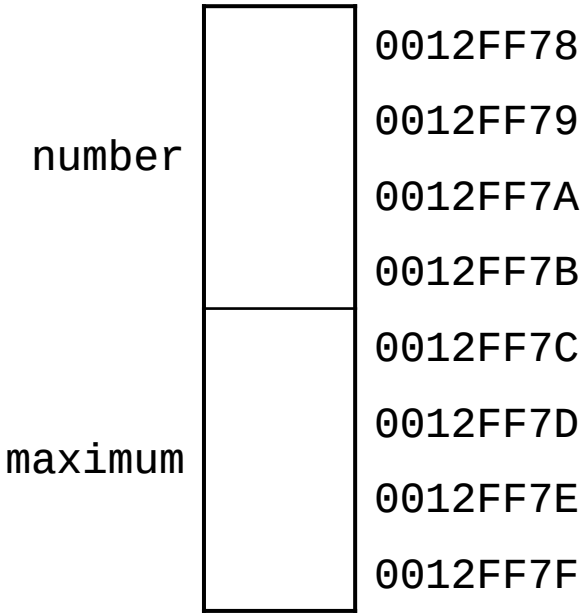
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1;        ;     )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

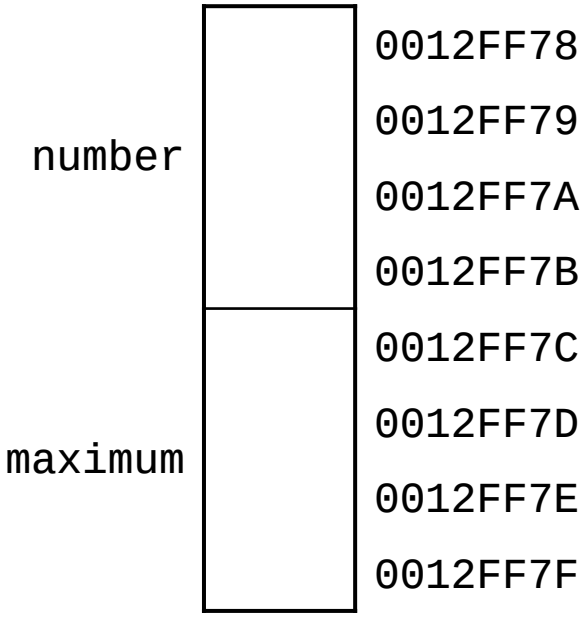
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1;        ; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

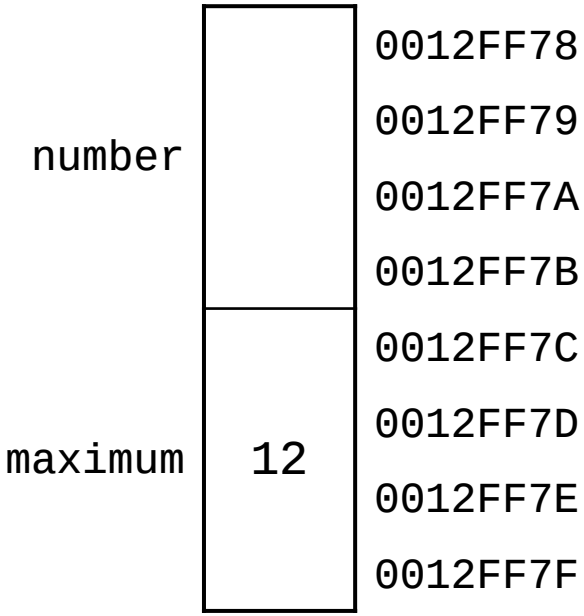
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

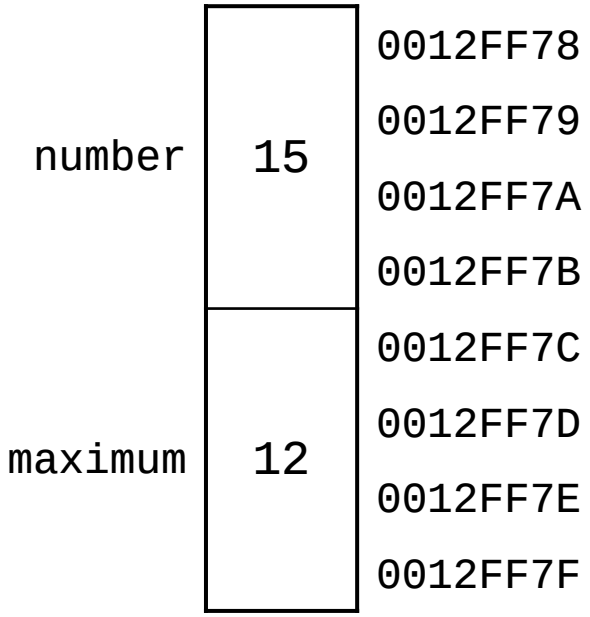
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

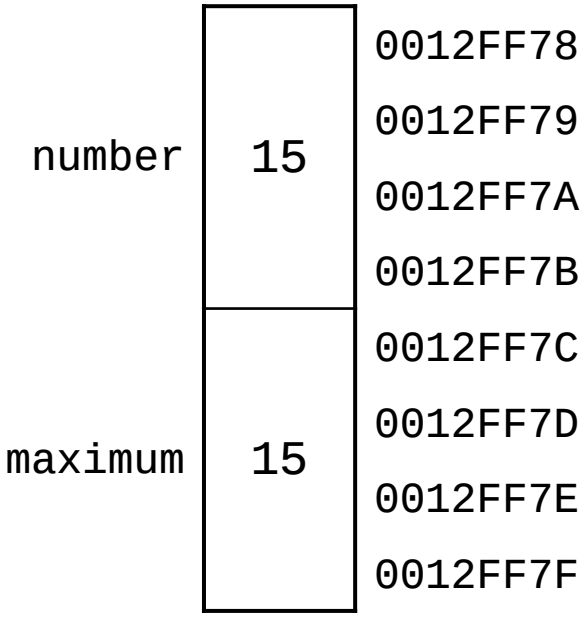
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

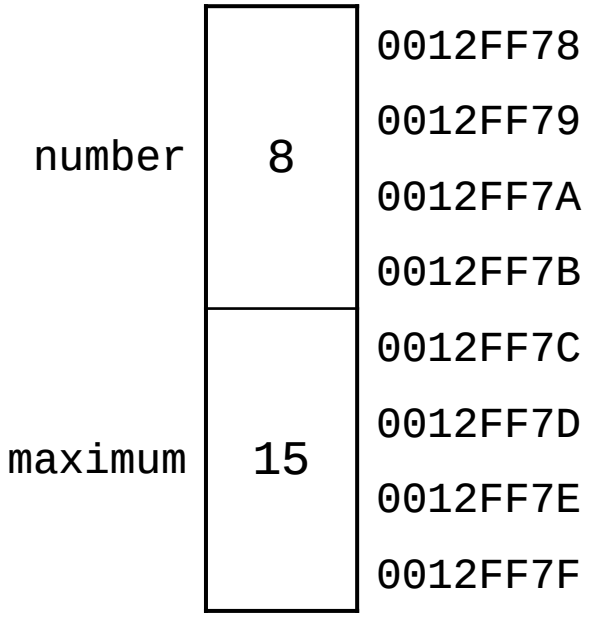
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

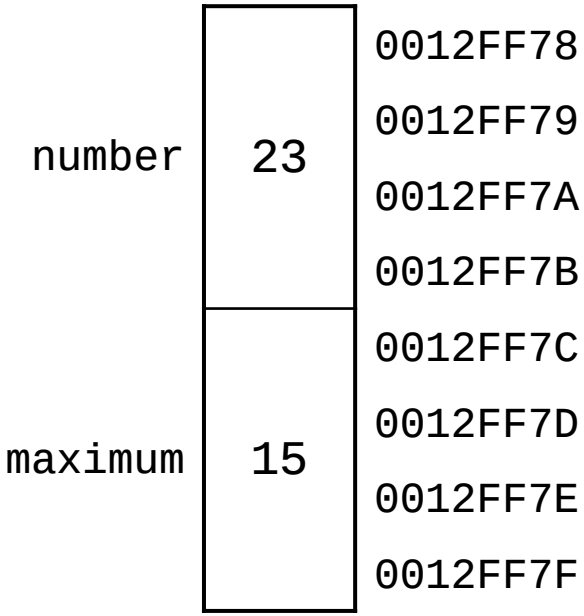
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

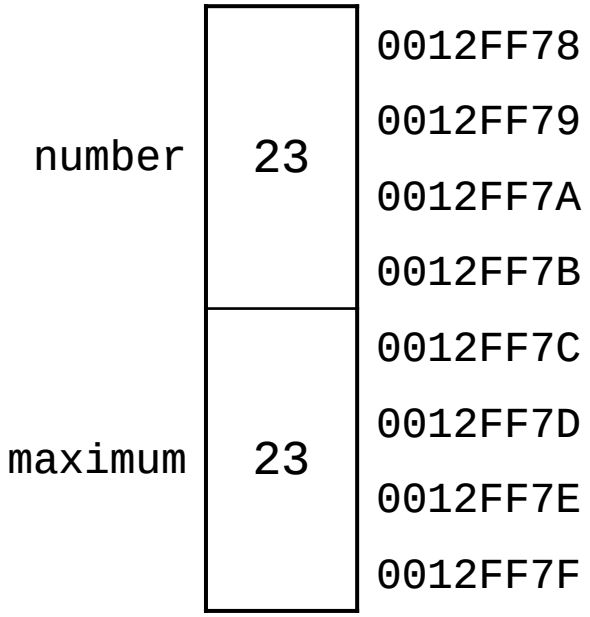
Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

Maximum Finding




```
cin >> maximum;
cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;

cin >> number;
if( number > maximum )
     maximum = number;
```

```
cin >> maximum;
for( i = 1; i <= 3; i++ )
{
   cin >> number;
   if( number > maximum )
      maximum = number;
}
```

Print a Triangle Pattern

Write a program that uses `for` statements to print the following pattern. All asterisks (`*`) should be printed by a single statement of the form `cout << '*';`

```
*
**
***
****
```

Print a Triangle Pattern

```
int main()
{
   for( int column{ 1 }; column <= 1; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 2; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 3; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 4; column++ )
      cout << "*";
   cout << endl;
}
```

Print a Triangle Pattern

```
int main()
{
   for( int          ;          ;        )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print a Triangle Pattern

```
int main()
{
   for( int row{ 1 };          ;        )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print a Triangle Pattern

```
int main()
{
   for( int row{ 1 };          ; row++ )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print a Triangle Pattern

```
int main()
{
   for( int row{ 1 }; row <= 4; row++ )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print Another Triangle Pattern

Write a program that uses `for` statements to print the following pattern. All asterisks (`*`) should be printed by a single statement of the form `cout << '*';`

```
****
***
**
*
```

Print Another Triangle Pattern

```
int main()
{
   for( int column{ 1 }; column <= 4; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 3; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 2; column++ )
      cout << "*";
   cout << endl;

   for( int column{ 1 }; column <= 1; column++ )
      cout << "*";
   cout << endl;
}
```

Print Another Triangle Pattern

```
int main()
{
   for( int        ;         ;       )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print Another Triangle Pattern

```
int main()
{
   for( int row = 4;          ;       )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print Another Triangle Pattern

```
int main()
{
   for( int row = 4;          ; row-- )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Print Another Triangle Pattern

```
int main()
{
   for( int row = 4; row >= 1; row-- )
   {
      for( int column{ 1 }; column <= row; column++ )
         cout << "*";
      cout << endl;
   }
}
```

Characters

```
ch1: 3
ch2: 5
ch1: 51
ch2: 53
ch1 + ch2: 104
```

Output

		0012FF40	
		0012FF41	
		0012FF42	
ch2	00110111	0012FF43	5
		0012FF44	
		0012FF45	
		0012FF46	
ch1	00110011	0012FF47	3

```
int main()
{
   char ch1 = '3';
   char ch2 = '5';
   cout << "ch1: " << ch1 << endl;
   cout << "ch2: " << ch2 << endl;
   cout << "ch1: " << static_cast< int >( ch1 ) << endl;
   cout << "ch2: " << static_cast< int >( ch2 ) << endl;
   cout << "ch1 + ch2: " << ch1 + ch2 << endl << endl;
}
```

Characters

```
ch1: +
ch2: -
ch1: 43
ch2: 45
ch1 + ch2: 88
```

Output

		0012FF40	
		0012FF41	
		0012FF42	
ch2	00101101	0012FF43	-
		0012FF44	
		0012FF45	
		0012FF46	
ch1	00101011	0012FF47	+

```
int main()
{
   char ch1 = '+';
   char ch2 = '-';
   cout << "ch1: " << ch1 << endl;
   cout << "ch2: " << ch2 << endl;
   cout << "ch1: " << static_cast< int >( ch1 ) << endl;
   cout << "ch2: " << static_cast< int >( ch2 ) << endl;
   cout << "ch1 + ch2: " << ch1 + ch2 << endl << endl;
}
```

ASCII character set										
	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	'	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

0	00000000	16	00010000	32	00100000	48	00110000	0
1	00000001	17	00010001	33	00100001	49	00110001	1
2	00000010	18	00010010	34	00100010	50	00110010	2
3	00000011	19	00010011	35	00100011	51	00110011	3
4	00000100	20	00010100	36	00100100	52	00110100	4
5	00000101	21	00010101	37	00100101	53	00110101	5
6	00000110	22	00010110	38	00100110	54	00110110	6
7	00000111	23	00010111	39	00100111	55	00110111	7
8	00001000	24	00011000	40	00101000	56	00111000	8
9	00001001	25	00011001	41	00101001	57	00111001	9
10	00001010	26	00011010	42	00101010	58	00111010	
11	00001011	27	00011011	43	00101011	59	00111011	
12	00001100	28	00011100	44	00101100	60	00111100	
13	00001101	29	00011101	45	00101101	61	00111101	
14	00001110	30	00011110	46	00101110	62	00111110	
15	00001111	31	00011111	47	00101111	63	00111111	

64	01000000		80	01010000	P	96	01100000		112	01110000	p
65	01000001	A	81	01010001	Q	97	01100001	a	113	01110001	q
66	01000010	B	82	01010010	R	98	01100010	b	114	01110010	r
67	01000011	C	83	01010011	S	99	01100011	c	115	01110011	s
68	01000100	D	84	01010100	T	100	01100100	d	116	01110100	t
69	01000101	E	85	01010101	U	101	01100101	e	117	01110101	u
70	01000110	F	86	01010110	V	102	01100110	f	118	01110110	v
71	01000111	G	87	01010111	W	103	01100111	g	119	01110111	w
72	01001000	H	88	01011000	X	104	01101000	h	120	01111000	x
73	01001001	I	89	01011001	Y	105	01101001	i	121	01111001	y
74	01001010	J	90	01011010	Z	106	01101010	j	122	01111010	z
75	01001011	K	91	01011011		107	01101011	k	123	01111011	
76	01001100	L	92	01011100		108	01101100	l	124	01111100	
77	01001101	M	93	01011101		109	01101101	m	125	01111101	
78	01001110	N	94	01011110		110	01101110	n	126	01111110	
79	01001111	O	95	01011111		111	01101111	o	127	01111111	

String

- Character constant
  - Enclosed in single quotes,
  - for example: `'z'`
- A string is a series of characters treated as a single unit.
  - May include letters, digits and various special characters such as `+`, `-`, `*`, `/` and `$`.
- A string is an array of characters ending with a null character (`'\0'`)
- String
  - Enclosed in double quotes,
  - for example: `"I like C++"`

String

- All strings end with null (`'\0'`)
- Character array

```
char color[ 5 ] = "blue";
```

  - Creates **5** element `char` array `color`
  - Null character (`'\0'`) implicitly added
- Alternative for character array

```
char color[ 5 ] = { 'b', 'l', 'u', 'e', '\0' };
```

	0	1	2	3	4
color	b	l	u	e	\0

color[0]	b	0012FF48
color[1]	l	0012FF49
color[2]	u	0012FF4A
color[3]	e	0012FF4B
color[4]	\0	0012FF4D

String

- All strings end with null (`'\0'`)
- Character array

```
char color[] = "blue";
```

  - Creates **5** element `char` array `color`
  - Null character (`'\0'`) implicitly added
- Alternative for character array

```
char color[] = { 'b', 'l', 'u', 'e', '\0' };
```

	0	1	2	3	4
color	b	l	u	e	\0

color[0]	b	0012FF48
color[1]	l	0012FF49
color[2]	u	0012FF4A
color[3]	e	0012FF4B
color[4]	\0	0012FF4D

String

- Input from keyboard

```
char color[ 10 ];
cin >> color;
```

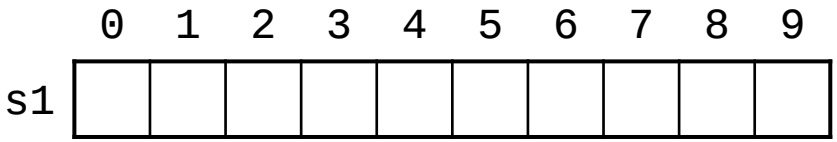
  - Puts user input in string
    - Stops at first whitespace character
    - Adds `null` character
  - If too much text entered, run time error
- Printing strings

```
cout << color << endl;
```

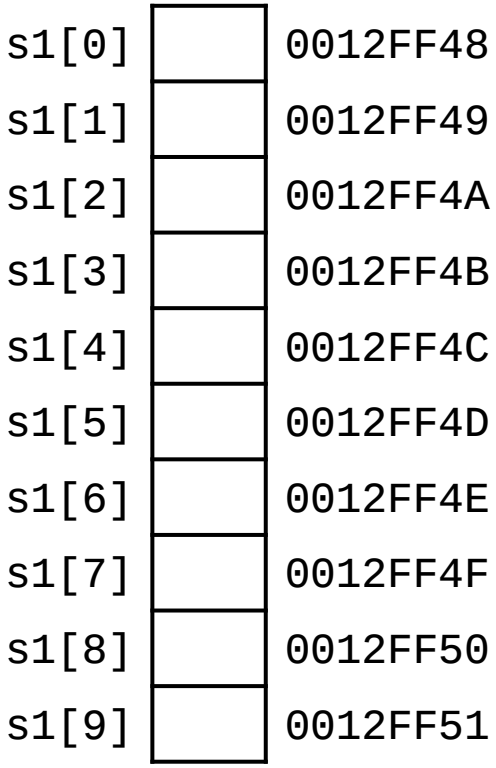
  - Does not work for other array types
  - Characters printed until **`null`** found

```
#include <iostream>
using namespace std;
int main()
{
   char string1[ 20 ];
   char string2[] = "happy new year";

   cout << "Enter the string \"merry christmas\": ";
   cin >> string1; // reads "merry"

   cout << "string1 is: " << string1 << "\nstring2 is: " << string2;

   cout << "\nstring1 with spaces between characters is:\n";
   for ( int i = 0; string1[ i ] != '\0'; i++ )
      cout << string1[ i ] << ' ';

   cin >> string1; // reads "christmas"
   cout << "\nstring1 is: " << string1 << endl;
}
```

```
Enter the string "merry christmas": merry christmas
string1 is: merry
string2 is: happy new year
string1 with spaces between characters is:
m e r r y
string1 is: christmas
```




```
int main()
{
   char s1[ 10 ];

   cin >> s1; // reads "merry"

   cout << "s1 is: " << s1;

   for( int i{ 0 }; s1[ i ] != '\0'; i++ )
      cout << s1[ i ] << ' ';

   cin >> s1; // reads "christmas"
   cout << "\ns1 is: " << s1 << endl;
}
```



	0	1	2	3	4	5	6	7	8	9
s1	m	e	r	r	y	\0				

```
int main()
{
   char s1[ 10 ];

   cin >> s1; // reads "merry"

   cout << "s1 is: " << s1;

   for( int i{ 0 }; s1[ i ] != '\0'; i++ )
      cout << s1[ i ] << ' ';

   cin >> s1; // reads "christmas"
   cout << "\ns1 is: " << s1 << endl;
}
```

s1[0]	m	0012FF48
s1[1]	e	0012FF49
s1[2]	r	0012FF4A
s1[3]	r	0012FF4B
s1[4]	y	0012FF4C
s1[5]	\0	0012FF4D
s1[6]		0012FF4E
s1[7]		0012FF4F
s1[8]		0012FF50
s1[9]		0012FF51

	0	1	2	3	4	5	6	7	8	9
s1	c	h	r	i	s	t	m	a	s	\0

```
int main()
{
   char s1[ 10 ];

   cin >> s1; // reads "merry"

   cout << "s1 is: " << s1;

   for( int i{ 0 }; s1[ i ] != '\0'; i++ )
      cout << s1[ i ] << ' ';

   cin >> s1; // reads "christmas"
   cout << "\ns1 is: " << s1 << endl;
}
```

s1[0]	c	0012FF48
s1[1]	h	0012FF49
s1[2]	r	0012FF4A
s1[3]	i	0012FF4B
s1[4]	s	0012FF4C
s1[5]	t	0012FF4D
s1[6]	m	0012FF4E
s1[7]	a	0012FF4F
s1[8]	s	0012FF50
s1[9]	\0	0012FF51

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
s2	h	a	p	p	y		n	e	w		y	e	a	r	\0

```
int main()
{
    char s2[] = "happy new year";

    cout << "s2 is: " << s2;
}
```

s1[0]	h	0012FF48
s1[1]	a	0012FF49
s1[2]	p	0012FF4A
s1[3]	p	0012FF4B
s1[4]	y	0012FF4C
s1[5]		0012FF4D
s1[6]	n	0012FF4E
s1[7]	e	0012FF4F
s1[8]	w	0012FF50
s1[9]		0012FF51
s1[10]	y	0012FF52
s1[11]	e	0012FF53
s1[12]	a	0012FF54
s1[13]	r	0012FF55
s1[14]	\0	0012FF56

Identification number

(1) 第一碼英文字母以下表轉換成數字

A	→ 10	台北市	J	→ 18	新竹縣	S	→ 26	高雄縣
B	→ 11	台中市	K	→ 19	苗栗縣	T	→ 27	屏東縣
C	→ 12	基隆市	L	→ 20	台中縣	U	→ 28	花蓮縣
D	→ 13	台南市	M	→ 21	南投縣	V	→ 29	台東縣
E	→ 14	高雄市	N	→ 22	彰化縣	W	→ 32	金門縣
F	→ 15	台北縣	O	→ 35	新竹市	X	→ 30	澎湖縣
G	→ 16	宜蘭縣	P	→ 23	雲林縣	Y	→ 31	陽明山
H	→ 17	桃園縣	Q	→ 24	嘉義縣	Z	→ 33	連江縣
I	→ 34	嘉義市	R	→ 25	台南縣			

Identification number

(1) 第一碼英文字母以下表轉換成數字

A	→ 10	台北市	J	→ 18	新竹縣			
B	→ 11	台中市	K	→ 19	苗栗縣	T	→ 27	屏東縣
C	→ 12	基隆市				U	→ 28	花蓮縣
D	→ 13	台南市	M	→ 21	南投縣	V	→ 29	台東縣
E	→ 14	高雄市	N	→ 22	彰化縣	W	→ 32	金門縣
F	→ 15	新北市	O	→ 35	新竹市	X	→ 30	澎湖縣
G	→ 16	宜蘭縣	P	→ 23	雲林縣			
H	→ 17	桃園市	Q	→ 24	嘉義縣	Z	→ 33	連江縣
I	→ 34	嘉義市						

Identification number

(2)　此數字之個位數乘以9再加十位數

(3)　第二碼到第九碼依次乘以8、7、・・・・、2、1

(4)　求(2)與(3)之和

(5)　若(4)為10的倍數則第十碼為0，否則，第十碼為10減[(4)除以10之餘數]

Identification number

(1) 第一碼英文字母以下表轉換成數字

A	→ 10	台北市	J	→ 18	新竹縣	S	→ 26	高雄縣
B	→ 11	台中市	K	→ 19	苗栗縣	T	→ 27	屏東縣
C	→ 12	基隆市	L	→ 20	台中縣	U	→ 28	花蓮縣
D	→ 13	台南市	M	→ 21	南投縣	V	→ 29	台東縣
E	→ 14	高雄市	N	→ 22	彰化縣	W	→ 32	金門縣
F	→ 15	台北縣	O	→ 35	新竹市	X	→ 30	澎湖縣
G	→ 16	宜蘭縣	P	→ 23	雲林縣	Y	→ 31	陽明山
H	→ 17	桃園縣	Q	→ 24	嘉義縣	Z	→ 33	連江縣
I	→ 34	嘉義市	R	→ 25	台南縣			

```
int firstChar[ 26 ] = { 10, 11, 12, 13, 14, 15, 16, 17, 34, 18,
                        19, 20, 21, 22, 35, 23, 24, 25, 26, 27,
                        28, 29, 32, 30, 31, 33 };
```

Identification number

(1) 第一碼英文字母以下表轉換成數字

A	→ 10	台北市	J	→ 18	新竹縣	S	→ 26	高雄縣
B	→ 11	台中市	K	→ 19	苗栗縣	T	→ 27	屏東縣
C	→ 12	基隆市	L	→ 20	台中縣	U	→ 28	花蓮縣
D	→ 13	台南市	M	→ 21	南投縣	V	→ 29	台東縣
E	→ 14	高雄市	N	→ 22	彰化縣	W	→ 32	金門縣
F	→ 15	台北縣	O	→ 35	新竹市	X	→ 30	澎湖縣
G	→ 16	宜蘭縣	P	→ 23	雲林縣	Y	→ 31	陽明山
H	→ 17	桃園縣	Q	→ 24	嘉義縣	Z	→ 33	連江縣
I	→ 34	嘉義市	R	→ 25	台南縣			

	0	1	2	3	4	5	6	7	8	9	10	11	12
firstChar	10	11	12	13	14	15	16	17	34	18	19	20	21

13	14	15	16	17	18	19	20	21	22	23	24	25
22	35	23	24	25	26	27	28	29	32	30	31	33

```
int main()
{
   int firstChar[ 26 ] = { 10, 11, 12, 13, 14, 15, 16, 17, 34, 18,
                           19, 20, 21, 22, 35, 23, 24, 25, 26, 27,
                           28, 29, 32, 30, 31, 33 };
   char id[ 11 ] = "";
   cout << "Enter the first 9 characters of an id number: ";
   cin >> id;

   int leadingNumber;
   if( id[ 0 ] >= 'A' && id[ 0 ] <= 'Z' )
      leadingNumber = firstChar[ id[ 0 ] - 'A' ];
   if( id[ 0 ] >= 'a' && id[ 0 ] <= 'z' )
      leadingNumber = firstChar[ id[ 0 ] - 'a' ];

   int sum = ( leadingNumber % 10 ) * 9 + ( leadingNumber / 10 );

   for( int i{ 1 }; i <= 8; i++ )
      sum += ( id[ i ] - '0' ) * ( 9 - i );

   if( sum % 10 != 0 )
      id[ 9 ] = 10 - ( sum % 10 ) + '0';

   cout << "\nThe corresponding id number is " << id << endl << endl;
}
```

```
int firstChar[ 26 ] = { 10, 11, 12, 13, 14, 15, 16, 17, 34, 18,
                        19, 20, 21, 22, 35, 23, 24, 25, 26, 27,
                        28, 29, 32, 30, 31, 33 };

if( id[ 0 ] >= 'A' && id[ 0 ] <= 'Z' )
   leadingNumber = firstChar[ id[ 0 ] - 'A' ];
```

id[0]	01000011	0012FF48	C	67
id[1]	00110001	0012FF49	1	49
id[2]	00110010	0012FF4A	2	50
id[3]	00110000	0012FF4B	0	48
id[4]	00110000	0012FF4C	0	48
id[5]	00110101	0012FF4D	5	53
id[6]	00110111	0012FF4E	7	55
id[7]	00110010	0012FF4F	2	50
id[8]	00110001	0012FF50	1	49
id[9]	00000000	0012FF51	\0	0
id[10]	00000000	0012FF52	\0	0

```
if( id[ 0 ] >= 'A' && id[ 0 ] <= 'Z' )
   leadingNumber = firstChar[ id[ 0 ] - 'A' ];
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
firstChar	10	11	12	13	14	15	16	17	34	18	19	20	21

13	14	15	16	17	18	19	20	21	22	23	24	25
22	35	23	24	25	26	27	28	29	32	30	31	33

id[0]	01000011	0012FF48	C	67
id[1]	00110001	0012FF49	1	49
id[2]	00110010	0012FF4A	2	50
id[3]	00110000	0012FF4B	0	48
id[4]	00110000	0012FF4C	0	48
id[5]	00110101	0012FF4D	5	53
id[6]	00110111	0012FF4E	7	55
id[7]	00110010	0012FF4F	2	50
id[8]	00110001	0012FF50	1	49
id[9]	00000000	0012FF51	\0	0
id[10]	00000000	0012FF52	\0	0

```
if( id[ 0 ] >= 'A' && id[ 0 ] <= 'Z' )
   leadingNumber = firstChar[ id[ 0 ] - 'A' ];
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
firstChar	10	11	12	13	14	15	16	17	34	18	19	20	21

13	14	15	16	17	18	19	20	21	22	23	24	25
22	35	23	24	25	26	27	28	29	32	30	31	33

id[0]	01000011	0012FF48	C	67
id[1]	00110001	0012FF49	1	49
id[2]	00110010	0012FF4A	2	50
id[3]	00110000	0012FF4B	0	48
id[4]	00110000	0012FF4C	0	48
id[5]	00110101	0012FF4D	5	53
id[6]	00110111	0012FF4E	7	55
id[7]	00110010	0012FF4F	2	50
id[8]	00110001	0012FF50	1	49
id[9]	00110011	0012FF51	3	51
id[10]	00000000	0012FF52	\0	0

break and continue Statements

```
int main()
{
   int i;
   bool isPrime = true;

   cout << "Enter an integer greater than 1: ";
   cin >> i;

   for( int j = 2; j < i; j++ )
      if(                 )
      {


      }

   cout << endl << i << " is ";
   if( !isPrime )
      cout << "not ";
   cout << "a prime number\n\n";
}
```

```
int main()
{
   int i;
   bool isPrime = true;

   cout << "Enter an integer greater than 1: ";
   cin >> i;

   for( int j = 2; j < i; j++ )
      if( ( i % j ) == 0 )
      {
         isPrime = false;
         break;
      }

   cout << endl << i << " is ";
   if( !isPrime )
      cout << "not ";
   cout << "a prime number\n\n";
}
```

```
#include <iostream>
using namespace std;

int main()
{
   for( int i = 1; i <= 20; i++ )
   {
      if( i % 3 == 0 )
         continue;

      cout << i << " ";
   }

   cout << "\nUsed continue to skip multiples of 3\n";
}
```

```
1 2 4 5 7 8 10 11 13 14 16 17 19 20
Used continue to skip multiples of 3
```

`switch` Multiple-Selection Statement

- The **`switch`** multiple-selection statement performs many different actions based on the possible values of a variable or expression.
- Each action is associated with the value of a constant integral expression (i.e., any combination of character and integer constants that evaluates to a constant integer value).

`switch` Multiple-Selection Statement

```
int main()
{
   int grade;
   int aCount = 0;
   int bCount = 0;
   int cCount = 0;
   int dCount = 0;
   int fCount = 0;

   cout << "Enter the letter grades.\n"
        << "Enter the EOF character to end input.\n";
```

```
while( ( grade = cin.get() ) != EOF )
{
   if( grade == 'A' || grade == 'a' )
        ++aCount;
   else if( grade == 'B' || grade == 'b' )
        ++bCount;
   else if( grade == 'C' || grade == 'c' )
        ++cCount;
   else if( grade == 'D' || grade == 'd' )
        ++dCount;
   else if( grade == 'F' || grade == 'f' )
        ++fCount;
   else if( grade != '\n' && grade != '\t' && grade != ' ' )
        cout << "Incorrect letter grade entered."
             << " Enter a new grade." << endl;
}
```

```
   cout << "\n\nTotals for each letter grade are:"
        << "\nA: " << aCount
        << "\nB: " << bCount
        << "\nC: " << cCount
        << "\nD: " << dCount
        << "\nF: " << fCount
        << endl;
}
```

```
grade = cin.get();
while( grade != EOF )
{
   if( grade == 'A' || grade == 'a' )
        ++aCount;
   else if( grade == 'B' || grade == 'b' )
        ++bCount;
   else if( grade == 'C' || grade == 'c' )
        ++cCount;
   else if( grade == 'D' || grade == 'd' )
        ++dCount;
   else if( grade == 'F' || grade == 'f' )
        ++fCount;
   else if( grade != '\n' && grade != '\t' && grade != ' ' )
        cout << "Incorrect letter grade entered."
              << " Enter a new grade." << endl;
   grade = cin.get();
}
```

```
#include <iostream>
using namespace std;

int main()
{
   system( "mode con cols=40 lines=22" );
   system( "color F0" );

   int grade; // letter grade entered by user
   int aCount{ 0 }; // counter of A grades
   int bCount{ 0 }; // counter of B grades
   int cCount{ 0 }; // counter of C grades
   int dCount{ 0 }; // counter of D grades
   int fCount{ 0 }; // counter of F grades

   cout << "Enter the letter grades." << endl
        << "Enter the EOF character to end input." << endl;
```

```
Enter the letter grades.
Enter the EOF character to end input.
a
B
c
C
A
d
f
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z

Number of students who received each letter grade:
A: 3
B: 2
C: 3
D: 2
F: 1
```

```
while ( ( grade = cin.get() ) != EOF )
{
   switch ( grade )
   {
      case 'A':
      case 'a':
         aCount++;
         break;
      case 'B':
      case 'b':
         bCount++;
         break;
      case 'C':
      case 'c':
         cCount++;
         break;
      case 'D':
      case 'd':
         dCount++;
         break;
      case 'F':
      case 'f':
         fCount++;
         break;
```

```
      case '\n':
      case '\t':
      case ' ':
         break;
      default:
         cout << "Incorrect letter grade entered."
              << " Enter a new grade." << endl;
         break;
   }
}

cout << "\n\nNumber of students who received each letter grade:"
     << "\nA: " << aCount
     << "\nB: " << bCount
     << "\nC: " << cCount
     << "\nD: " << dCount
     << "\nF: " << fCount
     << endl;
```

```
Enter the letter grades.
Enter the EOF character to end input.
a
B
c
C
A
d
f
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z

Number of students who received each letter grade:
A: 3
B: 2
C: 3
D: 2
F: 1
```

`switch` Multiple-Selection Statement

- The `cin.get()` function reads one character from the keyboard.
- Normally, characters are stored in variables of type `char`; however, characters can be stored in any integer data type, because types `short`, `int` and `long` are guaranteed to be at least as big as type `char`.
- Can treat a character either as an integer or as a character, depending on its use.

Operators Precedence & Associativity

Operator	Associativity
! ++ -- (post) ++ -- (pre) + - (unary)	Right to left
* / %	Left to right
+ - (binary)	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= *= /= %= += -=	Right to left